GTube: Geo-Predictive Video Streaming over HTTP in Mobile Environment

Jia Hao
School of Computing
National University of Singapore
Singapore 117417
haojia@comp.nus.edu.sg

Roger Zimmermann
School of Computing
National University of Singapore
Singapore 117417
rogerz@comp.nus.edu.sg

Haiyang Ma
Graduate School for Integrative Sciences and Engineering, National University of Singapore
haiyang@comp.nus.edu.sg

ABSTRACT

Mobile video streaming sometimes suffers from playback interruptions which are typically due to considerable network bandwidth variations that a user may experience when s/he travels along a route. Segmented adaptive HTTP streaming that switches between video streams encoded at different bitrates – and hence different quality levels – can be used to alleviate the issues of variable bandwidth. An important issue with respect to users' perceived experience is how the system schedules the quality levels to match the end-to-end network bandwidth capacity. It is very beneficial if the application can estimate the future network conditions in advance, and therefore perform quality control and buffer control wisely.

This work describes GTube, a video streaming system for receivers equipped with a GPS positioning sensor in a mobile environment. The available network bandwidth for each location is collected by mobile users and then this data along with the measured locations are uploaded and recorded in a server. Mobile devices that stream video can send queries to the server in order to better predict the near-future bandwidth availability and plan quality adaptations accordingly.

Keywords

Mobile video, Location services, Geo-prediction, DASH

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Application (multimedia streaming)*

1. INTRODUCTION

Mobile devices are popular for receiving video content on the go. As wireless connectivity is now commonly integrated into most handheld devices, streaming multimedia content to mobile peers is becoming a popular application that is increasingly available everywhere. Mobile data traffic, according to an annual report from Cisco Systems [6], continues to grow significantly. The forecast estimates that mobile video will generate over 66% of mobile data traffic by 2017.

Mobile media streaming may suffer from discontinuous playback which affects the user perceived Quality of Service (QoS). As the user of a portable device is moving along a path, network bandwidth fluctuates between different locations, and even in the same area the bandwidth can vary due to factors like the surrounding environment and the time of day. Most current mobile platforms do not predefine a strategy to deal with this situation. Hence how to handle this problem presents a challenge for mobile streaming applications.

Recently attention has focused on the Dynamic Adaptive Streaming over HTTP (DASH) standard [19], which has emerged as a popular video delivery mechanism based on HTTP progressive downloads. Its main features consist of (1) splitting a large video file into segments, (2) providing client-initiated flexible bandwidth adaptation by enabling stream switching among differently encoded segments, and (3) supporting near-live streaming events. DASH promises to improve users' end-to-end experience significantly by performing the video segment quality scheduling naturally and flexibly at the client side.

Under varying wireless conditions the configuration of a mobile media player's *quality scheduler* is hence both challenging and important. The general design goals of quality adaptation in adaptive HTTP streaming for mobile devices include the following aspects, in addition to the common rate adaptation for media streaming.

- The scheduler should identify whether a specific quality level matches the end-to-end network bandwidth capacity. It should utilize as much of the potential bandwidth as possible to provide the user with a high average video quality.
- The scheduler should have an effective buffer strategy to avoid streaming buffer underflows or overflows since such events cause playback interruptions.
- The scheduler should avoid rapid oscillations in quality, as this has an undesirable effect on the user's quality of experience.
- The scheduler should avoid changing the quality by too many levels in a single adjustment as it affects the

MMSys'14, March 19–21, 2014, Singapore, Singapore.

http://dx.doi.org/10.1145/2557642.2557647.

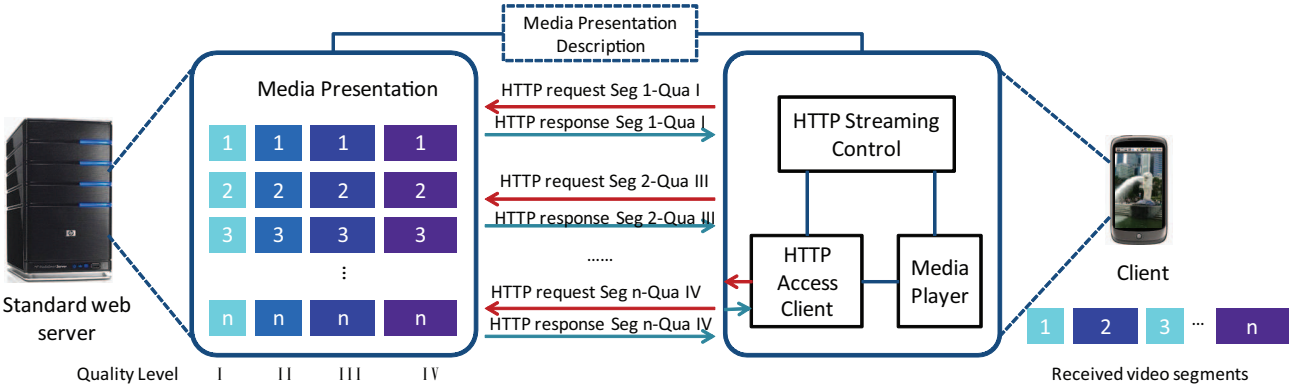


Figure 1: Dynamic Adaptive Streaming of HTTP (DASH) system.

user's perceived quality.

To satisfy the above design requirements, it is very beneficial if a streaming application can estimate the future network conditions in advance, therefore performing quality and buffer control smartly and efficiently. At least two key challenges exist for building such system. An initial and essential step is to obtain and maintain a bandwidth map, which relates the network condition to the location information. Applications that have access to such information will be able to anticipate upcoming fluctuations in the network before they occur. A reliable bandwidth prediction scheme which also needs to be carefully designed is the second major challenge.

Here we present a design framework for a *geo-predictive streaming system* called GTube to achieve these goals. The application obtains a user's location using the built-in Global Positioning System (GPS) receiver of a mobile device. A request indicating the device's trajectory will be sent to a server, and information about the expected connection quality in future locations is subsequently received. With this information, the application can adjust streaming parameters accordingly and provide a smooth and comparably high quality video stream, ensuring a good viewing experience for the user.

The contributions of our work are as follows. We have developed a smartphone application to gather information and relate it to GPS locations. The information collected is used to build the bandwidth map. A path prediction and a geo-based bandwidth estimation method is presented for estimating the future network conditions. Finally, we provide two quality adaptation algorithms which make use of the predicted bandwidth obtained in the previous step. Our simulation study results highlight that our approach can help streaming applications achieve fast and smooth adaptation to varying network conditions. The proposed scheme enables mobile clients to intelligently use location-specific bandwidth information in making quality adaptation decisions. Overall, the introduced solution achieves a balance between resource demands and quality of service.

The remainder of this paper is organized as follows. In Section 2, we introduce the background and summarize the related work. Section 3 describes the GTube framework. Section 4 describes several datasets we used, details the experimental setup and reports the experiment results. Finally Section 5 concludes the paper and presents future work.

2. BACKGROUND AND RELATED WORK

2.1 Adaptive HTTP Streaming Fundamentals

In the streaming media industry, HTTP-based media delivery has emerged as a de-facto streaming standard over recent years, replacing the existing media transport protocols such as RTP/RTSP. Despite its higher communication overhead and somewhat lack of technical finesse, its simple downloading model has attracted many content publishers, in part because it can reach a wide audience due to its high network penetrability and excellent match with existing HTTP-based caching infrastructures.

The 3GPP group recently standardized the adaptive HTTP streaming solution [19] as part of a Packet-switched Streaming Service (PSS) [20]. Adaptive HTTP streaming in 3GPP PSS follows a strategy of sequentially requesting and receiving small media chunks of the multimedia content, so-called media segments. 3GPP PSS adaptive HTTP streaming further enables the client to request media segments from different *representations* to react to varying network resources. Each *representation* consists of multiple media segments containing a certain duration of media data and encoded at a specific bitrate. Figure 1 shows a 3GPP adaptive HTTP streaming system. In the following paragraphs we briefly review several popular, commercial HTTP streaming solutions.

Adobe's HTTP Dynamic Streaming (HDS) is based on their MP4 fragment format (F4F) and its corresponding XML-based proprietary manifest file (F4M) [3]. HDS can be served from Flash Media Server (FMS) or an Apache server. The Adobe Flash Player is used on the client side to receive and render streams.

Microsoft's Smooth Streaming solution [23] is a compact and efficient method for the real-time delivery of MP4 files from the company's IIS web server, using a fragmented, MP4-inspired ISO/IEC 14496-12 ISO Base Media File Format specification [2]. The Smooth Streaming file format consists of a single physical file that encapsulates multiple self-contained "virtual" segments which can be retrieved through the use of a RESTful URL. This means that it requires a dedicated streaming web server that understands how to translate the URL requests into the corresponding byte offsets.

Apple has developed HTTP Live Streaming (HLS) [5] which uses an MPEG-2 Transport Stream (TS) as its delivery container format and an extension of the existing

proprietary MP3 playlist file format, *M3U8*, as its media descriptor container format. Nevertheless, Apple's solution has been widely supported by newer mobile devices and popular streaming platforms due to Apple's recent dominance in the smartphone and tablet markets.

2.2 Quality Adaptation Algorithm in Adaptive HTTP Streaming

There exists some prior work on the quality adaptation algorithm in HTTP streaming. Liu *et al.* [14] proposed an algorithm which compares the segment fetch time with the media duration contained in the segment to detect congestion and probe the spare network capacity. De *et al.* [8] proposed a Quality Adaptation Controller for live adaptive video streaming designed by employing feedback control theory. The controller tries to maintain the buffer level as stable as possible to match the video bitrate with the available bandwidth. QDASH [15] is a QoE-aware DASH system which takes into account the effect of the transition of quality levels on the QoE. Xiang *et al.* [21] focused on the quality adaptation algorithm for streaming scalable video in wireless networks. They formulated the quality adaptation problem as an Markov Decision Process (MDP) and used dynamic programming to solve the problem. Some research efforts provided evaluation on comparing the quality adaptation algorithms in the existing commercial players, and identified major differences between these adaptive media players in the market [16, 4].

For video quality adaptation, the existing HTTP streaming techniques only focus on the reaction to network conditions and player behavior. In contrast, our technique applies a predictive scheme to allow a more flexible operation.

2.3 Location-Aided Video Delivery System

Recently some video delivery systems associated with location information have been proposed [7, 18, 22, 17]. Curcio *et al.* [7] predicted network bandwidth for the future locations and adjusted the buffering parameters. However, they only provide a preliminary system design and their radio network throughput is modeled synthetical, while we use real-world traces. Singh *et al.* [18] proposed to build a Network Coverage Map Service to make rate-control decisions over RTP. Yao *et al.* [22] demonstrated the possibility to map network bandwidth to the road network through repeated measurements and revealed the strong correlation between WWAN bandwidth and location. More similar to our work, Riiser *et al.* [17] used real-world traces for predictive buffering in Dynamic Adaptive Streaming over HTTP (DASH). However, their prediction was based on the assumption that the server knows the the whole route information in advance (for example for a commuter), and they measured the bandwidth along that route for several times to get the average data. In contrast, our work does not make such assumptions, e.g., we do not know the future locations of the mobile user. We predict the path and calculate the bandwidth in 2-D geographical space.

3. GEO-PREDICTIVE VIDEO STREAMING SYSTEM

We now describe the design of GTube in details. Figure 2 shows a flowchart of the proposed system. As Geo-bandwidth Providers, mobile nodes collect required data

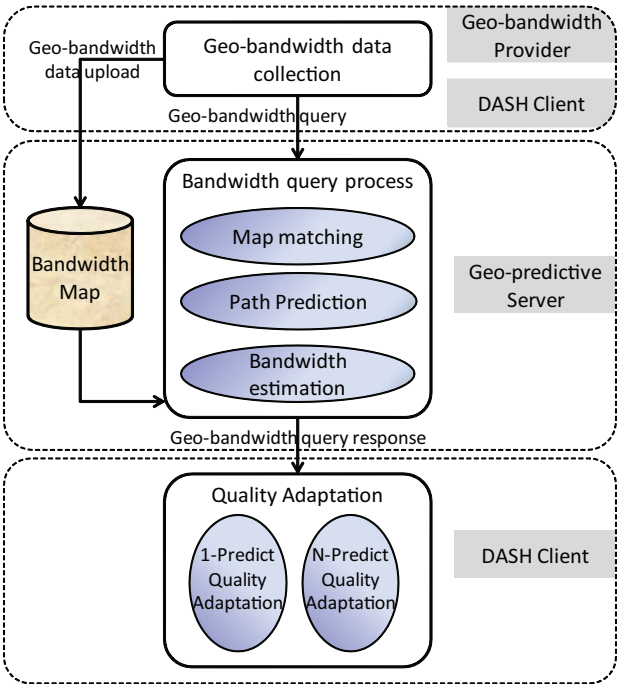


Figure 2: Flowchart of the proposed geo-predictive GTube streaming system.

tuples such as bandwidth, GPS location and capture time during video streaming. This functionality can be combined with a DASH player. The collected geo-bandwidth data are uploaded to the Geo-predictive Server immediately. The Geo-predictive Server is in charge of building and keeping a bandwidth map which correlates locations with other data associated to each location (e.g., network throughput, measurement time, etc.). In a typical streaming session, other users (DASH Clients) can request DASH segments from the Multimedia Server, and they are also responsible for sending their locations and speed to the Geo-predictive Server to obtain the predicted bandwidth. Upon receiving the bandwidth request, the Geo-predictive Server matches the current GPS sample point to the road network, predicts the future path of the client, then calculates the possible bandwidth (through a k-NN algorithm) along that path and sends it to the client. Based on the predicted bandwidth, the DASH Client applies the best quality level it can afford.

We first describe how the geo-bandwidth data are collected by the mobile client and uploaded to the server, and how a bandwidth request is issued. Next, we illustrate how the Geo-predictive Server estimates the bandwidth and acts to respond to the bandwidth request. Finally, we introduce two quality adaptation algorithms which both can utilize the bandwidth prediction scheme.

3.1 Geo-Bandwidth Data Collection and Upload

To build the bandwidth map, we need to record a large set of the locations (P) and bandwidth measurements (bw) of mobile clients. The collected meta-data streams are combined and analogous to sequences of $\langle nid, t, s, P, bw \rangle$ tuples, where nid represents the ID of the mobile device, t indicates

the time instant at which the data was recorded, and s is the speed of the client.

When a mobile client begins video streaming, the GPS is turned on to record the location as $\langle latitude, longitude \rangle$ pair. Our custom-written recording software receives location updates as soon as new values are available and records the updates along with the coordinated universal (UTC) time. The clients measure the receiver rate at each geo-location. Each recorded GPS update includes: (1) the current latitude and longitude, (2) the satellite time (in UTC) for the location update (in seconds), and (3) the current speed over ground. It is worth noting that this functionality can easily be added to any DASH client.

This part of the system also manages the storage of the bandwidth data on the device's flash disk. The goal is to utilize a flexible data format that can be easily ingested at a server. In this situation we choose JSON (JavaScript Object Notation) as the data interchange and storage format, since it has an equal descriptive power comparable to XML and has a much smaller grammar. The data format consists of four mandatory key attributes:

- **format_version**: The version number of the data format.
- **network_type**: The type of wireless internet access.
- **network_operator**: The provider of wireless communication services.
- **network_data**: Stores sequences of $\langle t, s, P, bw \rangle$ data collected from a mobile device.

Here is a sample specification of the data format that stores geo-bandwidth data.

```
{
  "format_version":"0.1",
  "network_type":"3G",
  "network_operator":"StarHub",
  "network_data":[
    {
      "location_array_timestamp_lat_long":[
        ["2013-03-18T07:58:41Z",1.29356,103.77],
        ["2013-03-18T07:58:46Z",1.29356,103.78]
      ]
    },
    {
       "speed_array_timestamp_s":[
        ["2013-03-18T07:58:41Z",10.20],
        ["2013-03-18T07:58:46Z",15.38]
      ]
    },
    {
       "bandwidth_array_timestamp_bw":[
        ["2013-03-18T07:58:41Z",2.103],
        ["2013-03-18T07:58:46Z",0.479]
      ]
    }
  ]
}
```

The client routinely sends POST HTTP requests with this information to the Geo-predictive Server. The data are uploaded to build the bandwidth map, which relates the network condition to geographic locations. A geo-bandwidth query is sent to the Geo-predictive Server together with these data, hence the server can perform bandwidth prediction and send the predicted values back to the DASH Client. Afterwards, the quality scheduler in the DASH Client will perform quality selection by using the predicted bandwidth.

3.2 Geo-Bandwidth Query and Response

The streaming client routinely queries the Geo-predictive Server for future bandwidth. It sends its current location and speed (g, s) to the server. If the user is traveling on a road, the Geo-predictive Server will perform map matching and path prediction for the mobile client along a road arc. Otherwise, the mobile client is assumed to move uniformly in a straight line and the future locations can be easily calculated. Then the server predicts the bandwidth for the future T time interval based on the historical bandwidth values measured around the predicted locations, and sends the list of estimated bandwidth values to the streaming client.

To explain our geo-bandwidth prediction scheme better, we now provide some definitions which will be used later.

Definition *(Road Network)*: The road network represents a finite street system. It consists of a set of one-way or two-way road curves in $\mathbb{R}^2$, each of which is called a road arc and assumed to be piecewise linear. An arc A can be completely characterized by a finite sequence of points $(a_1, a_2, \ldots, a_n)$, where the endpoints a_1 and a_n are referred to as nodes while $a_2, a_3, \ldots, a_{n-1}$ are referred to as shape points.

Definition *(Map Matching)*: Assume that a vehicle (or a person) is moving along a finite street system $\mathbb{N}$ and an abstract road network $\mathbb{N}'$ is used to represent this system. For this moving object, a sequence of observed positions of this object in the road network is acquired at a finite number of points in time, denoted by $t_1, t_2, ..., t_n$. This object's actual location at time t_n is denoted by v_n and the GPS sample point is denoted by g_n. Thus, map matching is to match the sample point g_n to an arc in the road network $\mathbb{N}'$, while determining the map-matched position on the arc that best corresponds to the vehicle's actual location v_n.

Definition *(Historical Trajectories)*: A user moving between two locations in a road network generates a trajectory. A user's trajectory is defined as a sequence of connected road arcs between two locations. GPS can record the user's locations at fixed time intervals and the speed at each location. These recorded locations together with speed information are transmitted to the server. On the server, these raw location sequences can be map-matched with the road network to generate a trajectory. The set of previous trajectories of a user are considered as his/her historical trajectories. In our study, trajectories are kept separate for each user, because at the crossroads different user may take different paths according to the user's moving preference. When a user is new to the target area, s/he does not have historical trajectories. In this case, the server uses a trajectory derived from the trajectories of all users.

Figure 3 illustrates the concept of bandwidth prediction. We first find the match point (red rectangle) of the GPS sample point (green diamond) on the road network (black line segment), then we predict the future locations (red circles). To narrow the scope, we issue a range query (blue dashed circle) with distance d and obtain the bandwidth measure points (blue triangles) within the range. Next the points whose sample time is within the adjacent time interval are selected. Among them we find the k nearest bandwidth

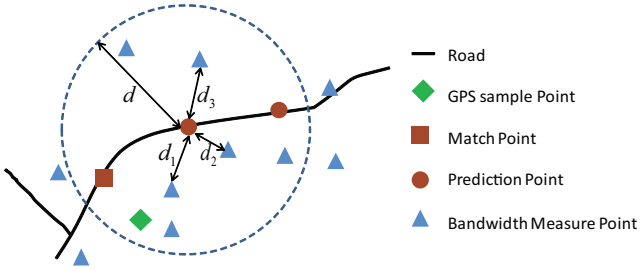


Figure 3: Illustration diagram of bandwidth prediction ($k = 3$).

measurement locations and finally we use the distance values d_1, d_2, d_3 as weight parameters to calculate the bandwidth.

We now describe the operation for bandwidth prediction in details. Assume that the user is located at point v on some road arc. Due to the inaccuracy of GPS, the obtained GPS sample point probably maps to a different location g.

- First, a map matching is performed for this new GPS sample point g. We find the most likely route which results in the trajectory data up to now. Then we select the last road arc A on this route as the most possible road arc the mobile user is traveling on now. Finally, we find the match point v of g on A such that it satisfies $v = argmin_{\forall v_i \in A} dist(v_i, g)$, where $dist(v_i, g)$ returns the distance between v_i and g.

  We adapt the improved HMM-based map matching algorithm introduced by Fang *et al.* [9]. To apply the HMM model, we can view the candidate road arcs in the road network as the hidden states, and the sample points derived from the noisy localization measurements as the observations. Then the map matching is redefined as finding the most probable arc sequence in the network which could have generated the given sample points. The algorithm utilizes the historical information from previous candidate arcs (candidate arcs of the previous sample point), as well as the topological information and speed constraints of the road network, instead of a range query, to find the current candidate arcs faster. Moreover, when it identifies a current candidate arc, only those previous candidate arcs from which this current candidate arc is temporally accessible are considered. Meanwhile the corresponding shortest routes are also found, respectively.

- Second, given the speed s of the mobile user and the match point v on road arc A obtained in the previous step, we aim to predict the future possible locations ($p \in P$) the user will be at. We assume a linear movement along the road arc. Denote T as the predicted time interval, and t as the time step. Hence, the future locations will be $p = v + s \times jt$, where j is an integer in the range of $[1, \lfloor \frac{T}{t} \rfloor]$. However, the above method only applies when the user moves along a single road arc. To handle the uncertainty problem when the user encounters an intersection, we adapt the predictive location model [12] to calculate the probability. We store the historical trajectories on the server and use them to infer the number of times the user has traveled on each road arc and the trajectory choice at each intersection. The model uses a probability matrix to describe the probabilistic information at an intersection. The matrix is dynamic and is updated periodically. The model evaluates all the options and selects the road arc with the highest probability.

- Third, since it is impossible to obtain bandwidth measurements for every single location, it is necessary to perform interpolation to obtain bandwidth estimations for target locations derived from adjacent points with bandwidth measurements. In our system we use the k-nearest neighbor algorithm (k-NN) to retrieve the nearest locations $(l_1, l_2, ..., l_k)$ and calculate the weighted mean bandwidth among them. For efficient processing, we issue a range query with distance d and gather the bandwidth measurement points within the range.

  As Figure 6 shows, the measured bandwidth for a single location can vary dramatically during the time of day, but the changing pattern tends to repeat itself over the days. That is, the bandwidth measurements in the same time period of different days shows little change. In order to make use of this property, we only select the bandwidth measurement points whose recording time is within the same time period (T_p) of the query time on different days. For example, assume the query time is 7am of day 3, given $T_p = 2h$, we will pick up all the measurement points whose recording time is between 6am and 8am of day 1 and day 2.

  Intuitively, we assume that the bandwidth values of near locations are more similar than those of locations farther apart. Therefore, we apply an Inverse Distance Weighted (IDW) interpolation for the bandwidth estimation. IDW assigns greater weights to points closer to the prediction location, and the weights diminish as a function of distance. The bandwidth value $B(p)$ of location p is determined by:

$$B(p) = \sqrt[pow]{\sum_{i=1}^{k} \left(\frac{B(l_i)}{dist(l_i, p)} \right)^{pow} / \sum_{i=1}^{k} \left(\frac{1}{dist(l_i, p)} \right)^{pow}}$$

  where *pow* is the power value that determines the rate at which the weights decrease.

Algorithm 1 outlines our method for bandwidth prediction. In line 1 function $MapMatching(G, g)$ finds the match point v of the GPS sample point g on a road network G. Then it calculates the future possible locations ($p \in P$) where the user will be at (lines 2–5). Next for all the predicted locations, $RangeQuery(p, d, L)$ uses location p as a central point and fetches points from set L within the requested range d. Function $TimeFilter(T_p, L_R)$ picks up the points whose sampling times are within T_p. $KNN(k, p, L_T)$ finds k nearest neighbors for location p among location set L_T, and the Euclidean distance is used as the distance metric. $IDW(L_k, B(L_k), p)$ uses the inverse weighted distance interpolation to calculate the estimated bandwidth value b at location p, and L_k is a set of known locations with bandwidth measurement values $B(L_k)$. Afterwards the estimated bandwidth value b is incorporated to the set B_p (lines 6–12). Finally the set of estimated bandwidth values is returned (line 13).

Algorithm 1 Algorithm for geo-bandwidth prediction

Require: g: Current GPS sample point
s: Current speed of the mobile client
T: Predict time interval
t: Time step
G: Road network
$M = (B, L)$: Bandwidth map. B–bandwidth, L–location
$B(l)$: Bandwidth measured in location l
T_p: Adjacent time period
b: Predict bandwidth value
Initialize $P = \emptyset$, where P is set of positions identified for predictive path
$B_p = \emptyset$, where B_p is a set of bandwidth values predicted for next T time

$v = MapMatching(G, g)$
for all $j \in [1, \lfloor \frac{T}{t} \rfloor]$ **do**
  $p = v + s \times jt$
  $P = P \cup p$
end for
for all $p \in P$ **do**
  $L_R = RangeQuery(p, d, L)$
  $L_T = TimeFilter(T_p, L_R)$
  $L_k = KNN(k, p, L_T)$
  $b = IDW(L_k, B(L_k), p)$
  $B_p = B_p \cup b$
end for
return B

3.3 Quality Adaptation

Upon receipt of the response of a previous bandwidth request, the DASH Client can make use of these data to perform quality adaptation. The quality adaptation algorithm determines the representation of the next media segment to be fetched each time after receiving the previous segment.

Suppose the stream we desire to transmit contains C representation levels. The levels are 0,1, ..., $C-1$ (0 is the index for the lowest level). r is the encoding media bitrate for the current video quality level used, and r' is the proposed level to be used after the next switch. Thus r_0 (r_{C-1}) represents the lowest (highest) bitrate. We write $r^-(r^+)$ for the bitrate of the next higher (lower) quality level with respect to r. Denote β as the current buffered media duration, β_{low} as the minimum buffered media duration required, and β_{high} as the maximum buffered media duration. We denote as ρ' the predicted bandwidth for the next time steps.

Based on the amount of available predicted bandwidth values, we propose two algorithms for quality adaptation.

A. 1-predict Algorithm

It is comparably simple and accurate if the Geo-predictive Server only needs to predict the bandwidth value for the next one time step. In this case, the quality adaptation algorithm can only take the most recent predicted bandwidth value as input and perform the quality selection. Here we propose the 1-predict rate adaptation algorithm which works as follows.

If $\beta < \beta_{low}$, there is a risk that the buffer will underflow, so switch to the lowest quality level.

While $\beta_{low} < \beta < \beta_{high}$, by comparing ρ' and r, we can make the switching decision as follows: (1) If $\rho' < r^-$, it means the bandwidth is too low to transmit the video segment at the current quality level, so decrease and switch to the first bitrate for which $r' < \rho'$. (2) If $r^- < \rho' \leq r$, then the bandwidth is not enough to transmit the video segment at the current quality level, but decreasing one level should be sufficient. (3) If $r < \rho' \leq r^+$, it means that the network condition is getting better, but switching up one level may cause the proposed bitrate to move higher than the average TCP throughput, so keeping the status unchanged is suitable for the current situation. (4) If $\rho' > r^+$, the available bandwidth can allow a higher video quality presentation level, so we can increase to r' which is the highest bitrate for which $r' < \rho'$.

If $\beta > \beta_{high}$, the client has quite a bit of buffered media to consume. In this case, we have two options: switching to a higher quality level, or keeping the current quality level and delay the transmission of the next video segment. If the current quality level is already the highest one, or the predicted bandwidth is less than the next higher bitrate, we will stay with the current quality level, and wait for a period of time until the buffer level reduces to a more moderate value ($\frac{\beta_{low}+\beta_{high}}{2}$). Otherwise, we will switch to a higher quality level.

Algorithm 2 1-predict quality adaptation algorithm

Require: r: Current bitrate used
r': Proposed bitrate
r_0: Lowest bitrate
r_{C-1}: Highest bitrate
ρ': Predicted bandwidth for next time interval
β: Current buffered media time
Initialize $r' = r, T_{idle} = 0$

if $\beta < \beta_{low}$ **then**
  $r' = r_0$
else if $\beta < \beta_{high}$ **then**
  if $r^- < \rho' \leq r$ **then**
    $r' = r^-$
  else if $r < \rho' \leq r^+$ **then**
    $r' = r$
  else
    $r' =$ the highest bitrate for which $r' < \rho'$
  end if
else
  if $r = r_{C-1}$ or $\rho' < r^+$ **then**
    $T_{idle} = \beta - \frac{\beta_{low}+\beta_{high}}{2}$
  else
    $r' = r^+$
  end if
end if
return r', T_{idle}

Algorithm 2 outlines our method for the 1-predict quality adaptation. As a drawback, if we only use the predicted bandwidth value in the next time step to determine the video rate since bandwidth is naturally time and location varying, the curve of the selected video rate over time will not be smooth. Furthermore, the user's quality of experience will suffer due to frequent video bitrate fluctuations. Hence, to avoid this phenomenon and to stabilize the video quality, we propose to use the *running average bandwidth* to replace the next predicted bandwidth value in Algorithm 2. For simplicity, we only use the 2 seconds running average connection throughput: the predicted bandwidth for this second and the actual bandwidth for the last second. Denote ρ_i as the actual bandwidth in the $i'th$ time interval and ρ_i' as the predicted bandwidth, then the 2-sec running average bandwidth is:

$$\overline{\rho_i} = \begin{cases} \lambda_1 \rho_i' + \lambda_2 \rho_{i-1} + (1 - \lambda_1 - \lambda_2)\overline{\rho_{i-1}} & i > 0 \\ \rho_0 & i = 0 \end{cases} \quad (1)$$

In the experiments, we set the same weight for ρ_i and ρ_i', $\lambda_1 = \lambda_2 = 0.3$.

B. N-predict Algorithm

If the Geo-predictive Server can estimate the bandwidth values for the next N time steps, the rate adaptation algorithm can utilize all this information and perform more smooth rate switching. Additionally it can ensure better bandwidth utilization.

Here we propose the N-predict quality adaptation algorithm. Denote $\rho_0, \rho_1, \ldots, \rho_{N-1}$ as the predicted bandwidth values for the next N time steps. The video data that can be transmitted during this period is $\sum_{i=0}^{N-1} \rho_i$, and the whole media data that can be consumed during this interval is $\sum_{i=0}^{N-1} \rho_i + \beta'$, where β' is the buffered media which can be consumed and be determined by:

$$\beta' = \begin{cases} \beta - \beta_{low} & \beta > \beta_{low} \\ 0 & \beta \leq \beta_{low} \end{cases} \quad (2)$$

Then the bitrate r for the next several video segments to be transmitted is established by calculating the average transmitted throughput γ as depicted by Equation 3.

$$\alpha \times (\sum_{i=0}^{N-1} \rho_i + \beta') = N \times \gamma \quad (3)$$

Here, α $(0 < \alpha < 1)$ is a configuration parameter used to ensure that there is enough data that can be transmitted with the selected quality level. For the next N time steps, we can switch to r which is the highest bitrate for which $r < \gamma$. During the actual transmission, there is a chance that the buffer will underflow because the selected bitrate is calculated based on an average value. To avoid this phenomenon, if $\beta < \beta_{low}$, we decrease the quality level by one to ensure smooth playback.

4. EVALUATION

To demonstrate the feasibility of our proposed GTube framework, we conducted experiments on the datasets we collected. In this section, we first introduce several datasets used for evaluation, and then illustrate the experimental setup. Finally, we report on the experimental results and provide some discussions.

4.1 Datasets

A. Road Network

Our road network information is based on OpenStreetMap [1] data. We extract the XML data from the map around the area of the National University of Singapore (NUS). The road network is contained within a rectangular area with the top left corner at (lat/long 1.306570, 103.765195) and bottom right corner at (lat/long 1.288520, 103.789485).

B. Bandwidth Traces

We collected bandwidth data of the NUS campus by traveling on school buses and by walking around the campus area. We used an HTC Nexus One phone to perform HTTP streaming downloads. The phone received the stream over its 3G interface and used the internal GPS module for location tracking. Our custom-written data acquisition software fetches information such as location, timestamp, etc., and calculates the average throughput every second as the observed bandwidth. We collected more than 1,000 bandwidth values in a 7-day period. Figure 4 shows some of the bandwidth traces around the NUS campus. This map is used by the Geo-bandwidth Server to predict the bandwidth.

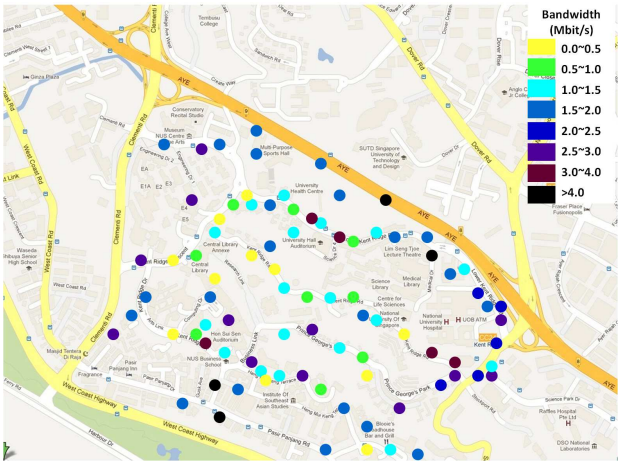


Figure 4: An example of a bandwidth map for the NUS campus.

Specifically, in our experiments we use two traces – Track 1 (184s) and Track 2 (283s) – to evaluate the performance of the quality adaptation algorithms. Their bandwidth values are shown in Figure 8(a) and Figure 9(a).

C. GPS Traces

To evaluate the accuracy of the map-matching and path prediction algorithm proposed in Section 3.2, we need a GPS trace dataset which contains historical trajectories and user identification information. To obtain these data, we employed 5 students to record their GPS trajectories in the NUS campus for a 7-day period. During that week, they performed their daily activities (e.g., go to the canteen on a school bus, walk to a lecture room or gym, etc.) as they usually did. The average total length of the GPS trace for one user is $18km$.

State-of-the-art mobile devices report the GPS accuracy of the received GPS signal in addition to the location. We found that the two different mobile devices we tested had two different minimum GPS error bounds: 6 meters and 2 meters. The GPS error distribution for our GPS trace dataset is shown in Figure 5. One can see that most of the GPS errors are below 20 meters.

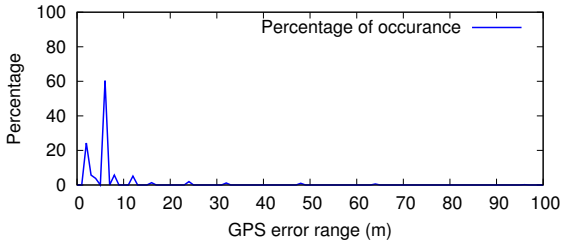


Figure 5: GPS error distribution for GPS trace dataset.

[1] http://www.openstreetmap.org/

Module	Parameter	Description	Values
Path Prediction	N	Predict time step	1–200, **10**
	t	Time step	**1s**
	$T = N \times t$	Predict time interval	1s–200s, **10s**
Bandwidth Prediction	T_p	Selected adjacent time period	1h–24h, **3h**
	pow	Power value used in IDW	**2**
	k	Input for k-NN algorithm	1–20, **5**
Quality Adaptation	β_{low}, β_{high}	Buffer setting	$\beta_{low} = 10s$, $\beta_{high} = 50s$
	λ_1, λ_2	Weight for running average bandwidth	$\lambda_1 = 0.3$, $\lambda_2 = 0.3$
	α	Configure parameter N-predict	**0.5**

Table 1: Parameters used in the experiments (values in bold are the default settings).

D. Video Segments

The used video sequence (Big Buck Bunny) was generated with an open source DASH content generation tool [13]. It has duration of approximately 600 seconds. The segment size is 2s. The available bitrates are 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.9, 1.2, 1.5, 2.0, 2.5, and 3.0 Mbit/s.

For later use, we note here that the smallest percentage difference between bitrates is $(0.7 - 0.6)/0.7 = 14.3\%$.

4.2 Experimental Setup

We implemented an extensive simulator by using C++ and the ns2 platform [11]. We assume an urban wireless communication infrastructure where mobile users are traveling on the road network of the NUS campus. The bandwidth prediction module is implemented with C++. The simulator took the measured bandwidth traces, the historical trajectories and the mobile user's current location and speed as input, and estimated the geo-bandwidth information for the user. The network condition is simulated by ns2, and we implemented a DASH client also in ns2, where various quality adaptation algorithms can be tested. The simulation parameters are summarized in Table 1.

4.3 Evaluation Metrics

To measure the performance of our proposed approach, we now describe the evaluation metrics we focus on for different modules.

A. Path Prediction

Correct location prediction rate: As Figure 5 shows, most of the GPS errors are below $20m$. If the distance between the predicted and actual location is less than $20m$, we consider the prediction to be correct. The correct location prediction rate is calculated as the ratio of the number of correct location predictions to the number of all location predictions.

B. Bandwidth Prediction

Correct bandwidth prediction rate: As we calculated in the video segments dataset, the smallest percentage bitrate differences between different video quality level is 14.3%. If the difference between the predicted and actual bandwidth is less than 14.3%, we consider the prediction to be correct. The rational behind this assumption is: this small shift will not cause the DASH client to switch the video quality for more than one level. The correct bandwidth prediction rate is calculated as the ratio of the number of correct bandwidth predictions to the number of all bandwidth predictions.

C. Quality Adaptation

For comparison, we implemented two other quality adaptation algorithms. The first is the algorithm proposed by Liu *et al.* [14], where the authors use $\mu = MSD/SFT$ as the transmission metric to decide whether to switch up or down (MSD is the media segment duration and SFT is the segment fetch time). If $\mu > 1 + max((b_{r_{i+1}} - b_{r_i})/b_{r_i})$, the chosen video rate will be switched up one level, where b_{r_i} denotes the encoded media bitrate of representation i. If $\mu < \gamma_d$ where γ_d denotes the switch down threshold, the chosen video rate reduces to a level to meet $b_{r_i} < \mu b_c$, where b_c denotes the bitrate of the current representation. To prevent a buffer overflow, before sending the next GET request, the algorithm will wait for $t_s = t_m - t_{min} - (b_c/b_{min})MSD$ seconds, where t_s, t_m and t_{min} denote the idle time in seconds, the buffered media time, and b_c and b_{min} denote the current representation bitrate and the minimum representation bitrate, respectively. The other algorithm implemented for comparison is the quality adaptation algorithm used by Adobe's Open Source Media Framework (OSMF) [1], shown in Algorithm 3. The download ratio (*playback time of last segment downloaded / amount of time it took to download that whole segment, from request to finish*) is compared to the switch ratio (*claimed rate of proposed quality / claimed rate of current quality*). The quality switch decision is made based on the comparison of the two ratio values.

We used the same experimental environment and the same

Algorithm 3 Quality adaptation algorithm in OSMF

Require: $t_{lastseg}$: Time of downloading the last segment
r: Current bitrate used
r': Proposed bitrate
r_0: Lowest bitrate
r_{C-1}: Highest bitrate
τ: Segment duration
$r_{download} \leftarrow \tau / t_{lastseg}$

```
if r_download < 1 then
    if r > r_0 then
        if r_download < r^-/r then
            r' = r_0
        else
            r' = r^-
        end if
    end if
else
    if r < r_{C-1} then
        while r' < r_{C-1} do
            r' = r'^+
            if r_download < r'/r then
                break
            end if
        end while
    end if
end if
```

simulation parameters for the four algorithms. We conducted the same experiments as for our algorithms, and the same curves were plotted for comparison.

The following performance metrics were used for the evaluation.

- *Achieved segment quality level* along the routes according to time.
- *Ratio of Bandwidth utilization* refers to the ratio of the bitrate of selected quality level to the actual bandwidth.
- *Rate of video quality level shift* refers to the number of quality switches divided by the number of segment switches.

$$\frac{N_{switch}}{N_{segment} - 1}$$

This is used to quantify the frequency of quality switches.

4.4 Experimental Results

A. Path Prediction Results

It is well known that channel conditions can change very quickly in a wireless environment. To investigate the temporal property of dynamic changing bandwidth, we measured the bandwidth for a single location for 7 days. Figure 6 shows how the measured bandwidth for a single location can vary dramatically during the time of a day, but the changing pattern tends to repeat on different days. That is, the bandwidth measurements during the same time period of different days shows little change. Based on this property, bandwidth measurement points in adjacent time periods of different days can be used as input for bandwidth prediction as described in the second point of Section 3.2.

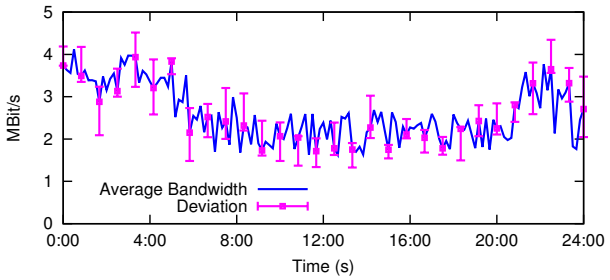


Figure 6: Bandwidth statistics for a single location at different times of 7 days.

Figure 7(a) shows the correct location prediction rate for different prediction time interval T values. When the prediction time becomes longer, more uncertainties will appear. For example, the user may speed up, slow down or stop. As a result the correct prediction rate decreases as T increases. In a later section, we will show the quality adaptation results for different T values.

B. Bandwidth Prediction Results

Next we investigate the impact of different parameters on the performance of the bandwidth prediction. Figure 7(b) shows the correct bandwidth prediction rate for different k values. When k is less than 3, the obtained input samples for bandwidth prediction are too few, resulting in a correct rate lower than 80%. On the other hand, when the k value is large (e.g., larger than 12), the prediction accuracy decreases, which is likely because too many noisy values are included into the calculation. Therefore, we set the default k value as 5, with the highest correct prediction rate of 94%.

Figure 7(c) shows the correct bandwidth prediction rate of different selected adjacent time period T_p values. When T_p is small, we do not have enough samples to obtain accurate prediction results. On the other hand, when the T_p value is large (e.g., larger than 15), the prediction accuracy is not satisfied either, which is probably because too many noisy values are taken into consideration. Hence we set the default T_p value as 3, which results in a comparably high correct prediction rate of 95%.

C. Quality Adaptation Results

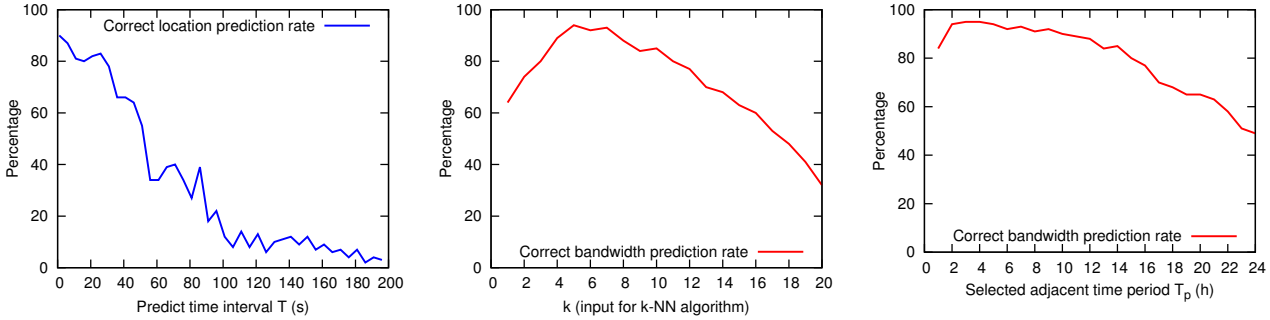
Figure 8 shows the bandwidth prediction and quality selection results for the four quality adaptation algorithms. Liu's algorithm uses conservative step-wise up switching and aggressive down switching (Figure 8(b)). The video quality is very stable, but the average video bitrate is very low. This is because the algorithm is too conservative and can not make full use of the bandwidth. In Adobe's algorithm (Figure 8(c)), the next segment to be downloaded is the one whose bitrate is closest to the recent bandwidth, with no considerations to stability or safety margins. As a result, the users' quality of experience suffers due to buffer underflow and too frequent oscillations in quality. To a certain extent, the 1-predict bandwidth (Figure 8(d)) is a smoothed version of the original TCP throughput (Figure 8(a)). Based on such a smoothed bandwidth trace, the 1-predict algorithm can achieve more stable quality switches compared to Adobe's algorithm. Furthermore, with the N-predict algorithm (Figure 8(e)), the video quality is smoothed out considerably as the algorithm can use enough predicted bandwidth in the calculation. It successfully achieves our goals: higher bandwidth utilization, avoiding switching too often, and avoiding the rapid fluctuations that are known to annoy users. Similar results can be seen from Figure 9 for Track 2.

Figure 10 presents the cumulative distribution function of the video data in each quality level for the four algorithms. For both tracks, the average quality level used in Liu's algorithm is much lower than the other three. The positive aspect is that it ensures the fluency of the playback. The CDF of the quality level is about the same for Adobe's and the 1-predict algorithm, while the performance of N-predict is slightly better than the other two. Especially, the amount of high quality level segments selected by 1-predict and N-predict are obviously more than for the other two, through which our algorithms can provide users a better viewing experience.

	Liu	Adobe	1-predict	N-predict
Track 1	60.1%	73.4%	85.6%	92.0%
Track 2	67.8%	78.5%	86.1%	93.3%

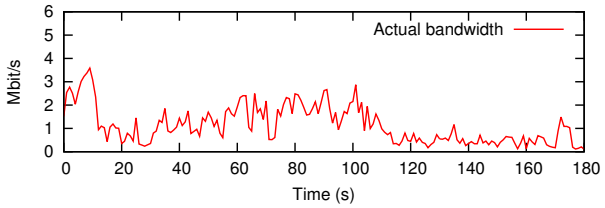
Table 2: Ratio of bandwidth utilization (higher numbers are better).

Table 2 shows the ratio of bandwidth utilization for the four algorithms. As we discussed above, the N-predict algorithm almost fully utilizes the available bandwidth (92.0%), and the ratio of bandwidth utilization for the 1-predict algorithm (85.6%) is higher than the other two.

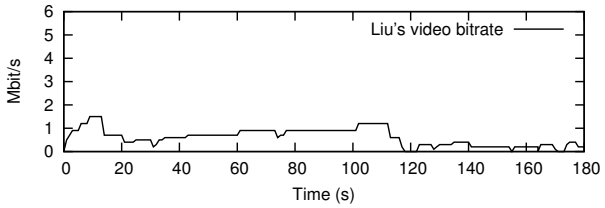


(a) Correct location prediction rate. (b) Correct bandwidth prediction rate. (c) Correct bandwidth prediction rate.

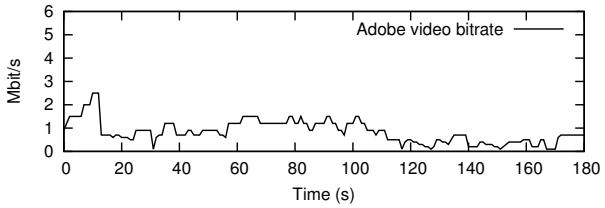
Figure 7: Evaluation results for path prediction and bandwidth prediction.



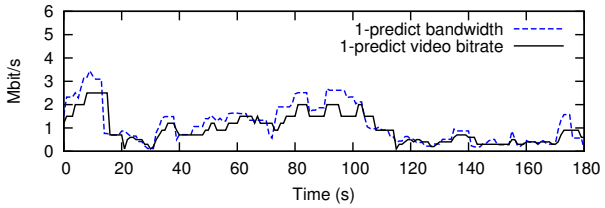
(a) Actual bandwidth.



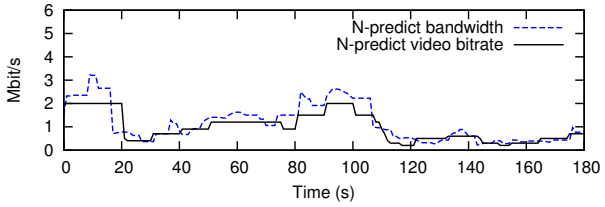
(b) Liu's algorithm.



(c) Adobe's algorithm.

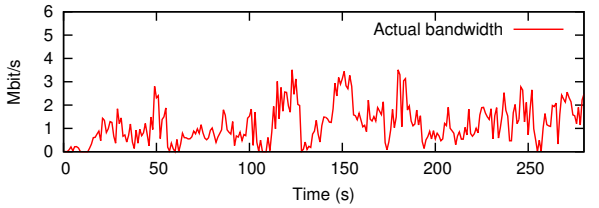


(d) 1-predict algorithm.

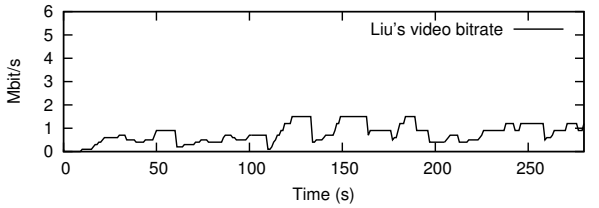


(e) N-predict algorithm. (N = 10)

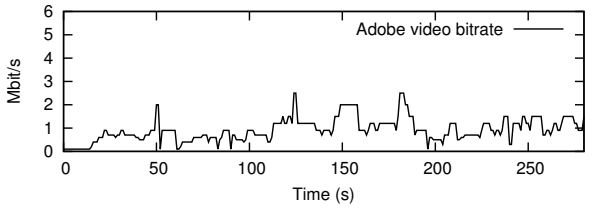
Figure 8: Video quality level for four algorithms for Track 1.



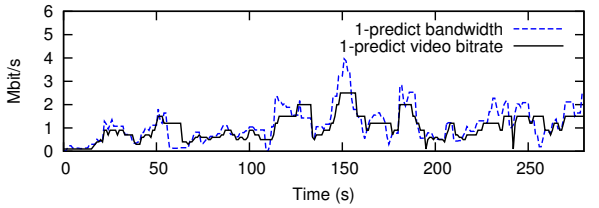
(a) Actual bandwidth.



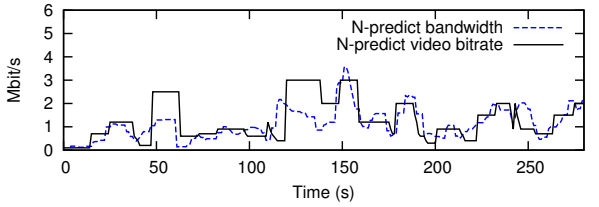
(b) Liu's algorithm.



(c) Adobe's algorithm.

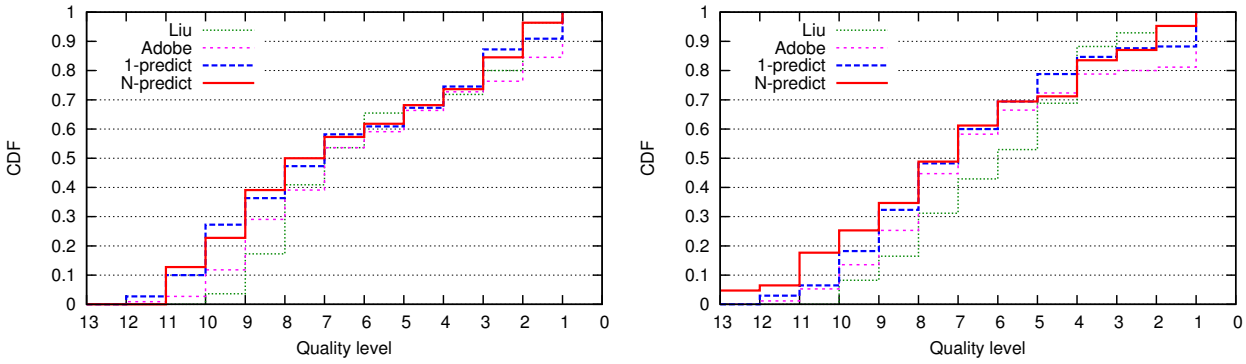


(d) 1-predict algorithm.



(e) N-predict algorithm. (N = 10)

Figure 9: Video quality level for four algorithms for Track 2.



(a) Track 1 (b) Track 2

Figure 10: Cumulative distribution function of quality (higher is better).

	Liu	Adobe	1-predict	N-predict
Track 1	31.4%	76.3%	50.8%	12.5%
Track 2	43.3%	72.9%	53.2%	11.9%

Table 3: Rate of video quality level shift (lower numbers are better).

In order to quantify the frequency of quality switching for each algorithm, we calculate the percentage of quality level switching by the number of quality switches divided by the number of segment switches ($\frac{N_{switch}}{N_{segment}-1}$). From Table 3, we can tell that the switch percentage of the 1-predict algorithm (50.8%) is lower than Adobe's algorithm (76.3%), but higher than Liu's algorithm (31.4%). While the N-predict algorithm (10.0%) performs better than the other three, making it the most stable scheduler that can provide a good viewing experience.

From the above observation, we find that the bandwidth prediction method is helpful to improve the stabilization of the video quality, and to increase the bandwidth utilization. We next turn our attention to the impact of the prediction time range on the N-predict algorithm. For different N values, Figure 11 shows the video quality level selection results for Track 2. As the prediction time increases, the stabilization of the streaming quality improves, i.e., it results in long intervals of the same quality. However, the chance of false bandwidth prediction also increases, and the chance of buffer underflow increases (210s in Figure 11(d) and 110s in Figure 11(e)). This reinforces the notion that more precise bandwidth detection results lead to better streaming quality.

4.5 Discussion

In our experiments we did not make any specific assumptions about the users' moving behavior and video viewing behavior. The robustness of our approach with respect to various anomalies is of course of significant importance. We have some preliminary indication that our method is quite robust. However, we will only have a clearer picture of these issues once we collect a much larger repository of data.

In our current implementation, we do not consider the battery life of mobile devices. For example, GPS is very energy-consuming. Energy-efficient data collection and transmission on mobile device have been discussed in many existing works [10, 9], which we will address in the future.

What we describe in this paper is a pipelined system with several components. To customize the service, one can replace the algorithm of each component with other algorithms. For example, different map-matching or path prediction methods could be used.

5. CONCLUSIONS

In this paper we have presented GTube, a design framework to enable geo-predictive mobile streaming. We have shown how to collect bandwidth data and build a bandwidth map, how to perform path and bandwidth prediction, and further how to use the prediction results to help improve the DASH streaming. We evaluated our scheme via experiments on real-world datasets, and compared it with two other algorithms. Our experiments show that the technique is effective to achieve continuous playback and to provide higher and stabilized media quality to the end user.

For future work we plan to extend our approach in several aspects. First, due to the power-consuming property of mobile streaming, it is necessary to provide energy-efficient streaming solutions for mobile devices. We plan to take the battery life into consideration for the geo-bandwidth data collection, upload and quality adaptation module. Second, we will investigate the problem of the frequency of geo-bandwidth data uploads and the frequency of bandwidth map updates, as they will affect the interaction between server and clients and further affect the users' perceived experience. Nevertheless, we expect our approach of leveraging location information to facilitate efficient mobile video delivery to be useful for a wide range of novel applications.

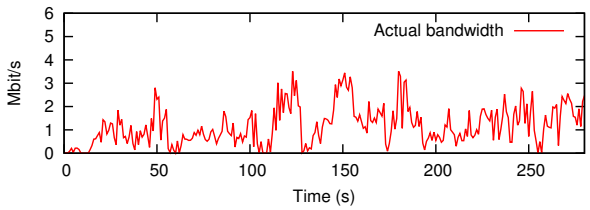
6. REFERENCES

[1] Open Source Media Framework (OSMF). `http://www.osmf.org`.

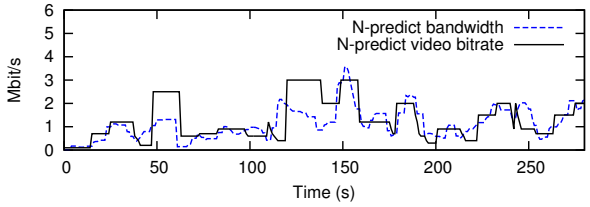
[2] Information Technology – Coding of Audio-Visual Objects – Part 12: ISO base Media File Format; ISO/IEC 14496-12. , 2009-07-29.

[3] Adobe. HTTP Dynamic Streaming on the Adobe Flash Platform. 2010.

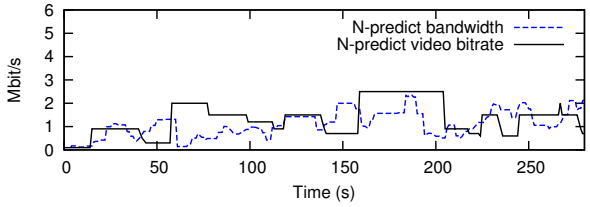
[4] S. Akhshabi, A. Begen, and C. Dovrolis. An Experimental Evaluation of Rate-adaptation Algorithms in Adaptive Streaming over HTTP. In *2nd annual ACM Conference on Multimedia Systems*, pages 157–168, 2011.



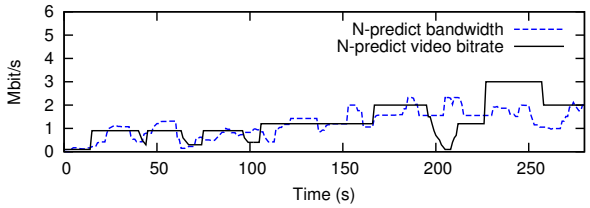
(a) Actual bandwidth.



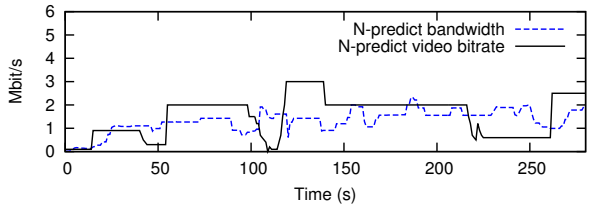
(b) N = 10



(c) N = 20



(d) N = 30



(e) N = 40

Figure 11: Video quality level for different N values obtained from the N-predict algorithm (Track 2).

[5] Apple Inc. HTTP Live Streaming draft-pantos-http-live-streaming-06. *Internet-Draft*, 2011.

[6] Cisco Systems, Inc. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2012-2017. White Paper, 2012.

[7] I. D. Curcio, V. K. M. Vadakital, and M. M. Hannuksela. Geo-Predictive Real-Time Media Delivery in Mobile Environment. In *3rd workshop on Mobile video delivery*, pages 3–8, 2010.

[8] L. De Cicco, S. Mascolo, and V. Palmisano. Feedback Control for Adaptive Live Video Streaming. In *2nd annual ACM conference on Multimedia systems*, pages 145–156, 2011.

[9] S. Fang and R. Zimmermann. EnAcq: Energy-Efficient GPS Trajectory Data Acquisition based on Improved Map Matching. In *19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 221–230, 2011.

[10] J. Hao, S. Kim, S. Ay, and R. Zimmermann. Energy-efficient Mobile Video Management using Smartphones. In *2nd annual ACM Conference on Multimedia Systems*, pages 11–22, 2011.

[11] Information Sciences Institute, The University of Southern California. The Network Simulator - ns-2, 2006.

[12] H. A. Karimi and X. Liu. A Predictive Location Model for Location-based Services. In *11th ACM International Symposium on Advances in Geographic Information Systems*, pages 126–133, 2003.

[13] S. Lederer, C. Müller, and C. Timmerer. Dynamic Adaptive Streaming over HTTP Dataset. In *3rd annual ACM Conference on Multimedia Systems*, pages 89–94, 2012.

[14] C. Liu, I. Bouazizi, and M. Gabbouj. Rate Adaptation for Adaptive HTTP Streaming. In *2nd annual ACM Conference on Multimedia Systems*, pages 169–174, 2011.

[15] R. K. Mok, X. Luo, E. W. Chan, and R. K. Chang. QDASH: a QoE-aware DASH System. In *3rd annual ACM Conference on Multimedia Systems*, pages 11–22, 2012.

[16] H. Riiser, H. Bergsaker, P. Vigmostad, P. Halvorsen, and C. Griwodz. A Comparison of Quality Scheduling in Commercial Adaptive HTTP Streaming Solutions on a 3G Network. In *4th Workshop on Mobile Video*, pages 25–30, 2012.

[17] H. Riiser, T. Endestad, P. Vigmostad, C. Griwodz, and P. Halvorsen. Video Streaming using a Location-based Bandwidth-Lookup Service for Bitrate Planning. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 8(3):24, 2012.

[18] V. Singh, J. Ott, and I. D. Curcio. Predictive Buffering for Streaming Video in 3G Networks. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012 IEEE International Symposium on a*, pages 1–10, 2012.

[19] T. Stockhammer. Dynamic Adaptive Streaming over HTTP: Standards and Design Principles. In *2nd annual ACM Conference on Multimedia Systems*, pages 133–144, 2011.

[20] V. Varsa and I. Curcio. Transparent End-to-End Packet Switched Streaming Service (PSS): Protocols and Codecs. Technical report, 3GPP TR 26.937 V1, 2003.

[21] S. Xiang, L. Cai, and J. Pan. Adaptive Scalable Video Streaming in Wireless Networks. In *3rd annual ACM Conference on Multimedia Systems*, pages 167–172, 2012.

[22] J. Yao, S. S. Kanhere, and M. Hassan. Improving QoS in High-Speed Mobility using Bandwidth Maps. *Mobile Computing, IEEE Transactions on*, 11(4):603–617, 2012.

[23] A. Zambelli. IIS Smooth Streaming Technical Overview. *Microsoft Corporation*, 2009.