Predictive Buffering for Streaming Video in 3G Networks

Varun Singh†, Jörg Ott†, Igor D. D. Curcio‡
†*Comnet, Aalto University, Espoo, Finland*
‡*Nokia Research Center, Tampere, Finland*
{*varun,jo*}*@comnet.tkk.fi, igor.curcio@nokia.com*

***Abstract*—This paper presents a multimedia streaming service in a mobile (3G) environment that, in addition to in-band congestion signals such as packet losses and delay variations, receives congestion cues from a Network Coverage Map Service (NCMS) to make rate-control decisions. The streaming client routinely queries the NCMS to assess the network conditions at future locations along its expected path. The streaming client may ask the streaming server for short-term transmission bursts to increase pre-buffering when it is approaching areas with bad network performance to maintain media quality. If needed, the client may also switch to a different encoding rate (*rate-switching*) depending on the severity of expected congestion. These notifications are scheduled as late as possible, so that any changes in network conditions and/or changes in user's movements can be taken into account (*late scheduling*). Using this type of geo-predictive media streaming service we show that the streaming client can provide pause-less playback and better quality of experience to the user.**

***Keywords*-Mobile Multimedia Streaming; Geo-prediction; Media Adaptation; Protocol Design;**

I. Introduction

Consumption of multimedia on mobile devices is increasing; YouTube[1] alone accounts for 22% of the mobile data bandwidth [2]. Moreover, the video streaming services cannot only be accessed by smartphones and tablets but also by constrained devices like a feature phones and in-vehicle entertainment systems. In a 3G network, mobility, fading, interference, cell loading, handovers and other factors can affect the throughput available to each user. Generally, video streaming applications overcome congestion by pre-buffering content. However, the video will pause if the buffer runs out and will not resume until a sufficient amount of video is pre-buffered. Usually, the streaming clients assume that the congestion will occur only for a few seconds and pre-buffer fixed amount of media data to overcome it, but in 3G networks loss in connectivity is unpredictable and outages can occur for a longer period of time. In these cases the stream suffers from repeated pausing and re-buffering, which degrades the perceived quality of the video content. One alternative would be to increase the size of the pre-buffer. But if the user skips to another video, then the extra pre-buffering will be useless. Another alternative would be to switch the media stream to a lower encoding rate and switch back later. However, the implementer should bear in mind not to change the media rate too often and not to vary the rate by a large amount as both these actions adversely affect the user's quality of experience [3].

Modern mobile networks have been designed to carry multimedia streams with QoS traffic classes [4], but deployments of GPRS, 3G and HSDPA networks show that there are still geographical areas where best-effort traffic classes are used [5], [6]. These constrained geographical areas may occur due to fading and interference from large building structures or closed or inaccessible areas (e.g., tunnels, boats on lakes or in the archipelago, rural areas).

In this paper, we propose an architecture (Section III) that enables the streaming client to predictively pre-buffer multimedia data based on the input from a Network Coverage Map Service (NCMS). To provide the streaming client with coverage notifications, the users share their current network characteristics and geo-location with the NCMS. We also describe in detail the signaling between the components (Section V), namely, the streaming server, streaming clients and NCMS. Additionally, we also propose methods for seeking out areas (*lookahead*) with poor connectivity (*coverage hole*) to schedule either switching to a lower media encoding rate (*rate-switching*) for the duration of poor coverage or increasing the media transmission rate before entering the area of poor coverage so that adequate media is buffered to overcome the outage (Section IV). This predictive method of pre-buffering is orthogonal to existing methods that react to congestion events by sensing changes in network characteristics.

II. Related Work

Early papers [7], [8], [9] for congestion control in multimedia over RTP are limited to Internet applications. With the emergence of 3G networks, congestion control needs to be reworked because the varying network capacity affects the video quality [10]. This problem has been the focus for 3GPP in the recent years (e.g. [11], [12]). A streaming client using Scalable Video Coding [13] and Multiple Description Coding [14], can prefetch the layers that do not exceed the currently available channel capacity. However, changing the rate too often or by too much may adversely affect the users' quality of experience [3]. Increasingly, web-services run variants of HTTP over TCP for media delivery [15], [16], [17] but TCP can only be used within strict bounds of

[1]http://m.youtube.com uses RTSP instead of HTTP for streaming [1]

978-1-4673-1239-4/12$31.00

network characteristics [18] which may be difficult to adhere to in a 3G environment.

Service Maps are introduced in [19] and the measurement based approach is proposed in [20]. CARS [21] evaluates rate-adaptation in a vehicular network and uses signal strength and throughput at a location as an indicator for congestion. GPS-based congestion control is introduced in [22], and evaluated in different scenarios [23], [24]. These papers take signal strength as an influential factor for rate-control and show that predicting based on signal strength alone is insufficient. Yet, they do show that using past information to predict future network characteristics is possible.

Preliminary analysis of a system similar to the one proposed in this paper is done in [5] but it is based on simulations, while this paper presents results from real-world experimentation. [6] also proposes a similar architecture (*bandwidth lookup service*) to the one in this paper and uses real-world traces to evaluate different types of averaging algorithms for predictive buffering in Dynamic Adaptive Streaming over HTTP (DASH). While the averaging algorithm is not a focus for this paper, we use K-means [25] and K-nearest neighbor (K-NN) [26] algorithms to form regions with similar bandwidth. Additionally, [6] proposes fetching the bandwidth along a travel route in steps of 100 meters, which we find limiting. Instead we propose multiple methods for discovering areas with poor connectivity and show that not only looking up future bandwidth but also when to vary the sending rate (*prefetch*) affects the usefulness of the service.

III. System Model

In this section, we introduce the different entities that make up the Geo-location Assisted Streaming System (GLASS) architecture and also briefly describe the interactions between the components.

A. Components

Figure 1 shows the GLASS architecture that mainly comprises of the following components: the streaming server, a Network Coverage Map Server (NCMS) and streaming clients.

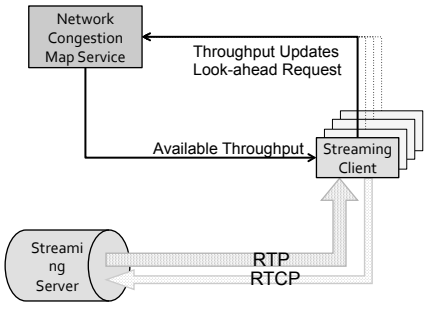


Figure 1: Shows the components in the GLASS Architecture. NCMS is the entity that keeps track of the throughput at each location and sends throughput recommendations to the streaming client.

The **Streaming Server** delivers multimedia content to a media player using RTP/UDP[2]. It is able to vary the transmission time of each packet and additionally is also able to switch between different pre-encoded video files (*rate-switching*).

The **Streaming Client** initiates, receives and plays back multimedia content from a streaming server. It must also support media delivery using RTP and advanced features of audio-video profile, such as early feedback, shorter and immediate RTCP feedback interval, NACK, etc. Furthermore, the streaming client is able to provide more precise rate-control based on the input from the Network Coverage Map Server (NCMS). The client should also be capable of increasing the available receiver buffer to handle more media data. In the context of this paper, the streaming client is running on a GPS-capable mobile device.

The **Network Coverage Map Server (NCMS)** receives location specific bandwidth updates from streaming clients (crowd-sourcing), creates bandwidth maps, sends bandwidth updates to clients for specific future locations. Unlike the interaction between the streaming client and server (over UDP), the streaming client communicates with the NCMS using HTTP POST and GET messages. The NCMS stores the location updates in a database and clusters the location updates based on the bandwidth and GPS accuracy parameters.

B. Terminology

Coverage Update: The mobile client measures and reports the operator name, base station ID (BSSID), signal strength, and GPS information (lat, lon, speed, trajectory).

Bandwidth: We loosely use the term to refer to the available capacity or end-to-end throughput. However, this depends on the type of information measured and analyzed by the streaming client. It can be current media receive rate, overall system throughput or something else. calculating it

Coverage Holes (T_{gap}): Areas where the bandwidth drops below the current media rate. T_{gap} is the duration of the coverage hole in seconds and depends on the user's traveling speed. The streaming client needs to pre-buffer the difference in the average media rate and available throughput to continuously play back media during the coverage hole.

C. Concept

The streaming client looks ahead at locations in its vicinity for bad coverage. The range of the lookahead depends on the speed and current buffering. When the streaming client discovers a coverage hole (T_{gap}), it calculates the time available to buffer the media content (*time-to-outage*, T_{tto}) based on the available throughput between its current position and the location of the coverage hole and the user's average vehicular speed. The streaming client attempts to

[2]In this paper, we focus on RTP for streaming video but nothing in the architecture prohibits the use of HTTP for streaming.

minimize the T_{tto} so that it does not unnecessarily buffer content if the travel route changes. Furthermore, if there is insufficient time to buffer the media content, the streaming client switches the media rate to a lower rate less than or equal to the throughput available in the coverage hole. Figure 2 depicts these steps for predictive buffering.

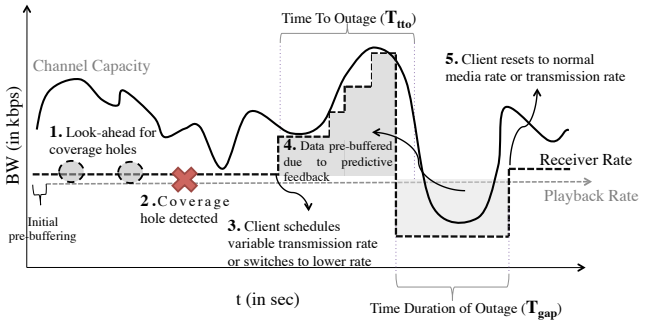


Figure 2: Shows how the predictive buffering concept works

D. Signaling

The interaction between the components can be broken down into the following steps, shown in Figure 3.

- **Coverage Updates**: Building the coverage map by getting updates on network conditions from the users.
- **Lookahead**: Fetching the upcoming network conditions from the map server.
- **Congestion Control**: Controlling media transmission rate based on the coverage map.

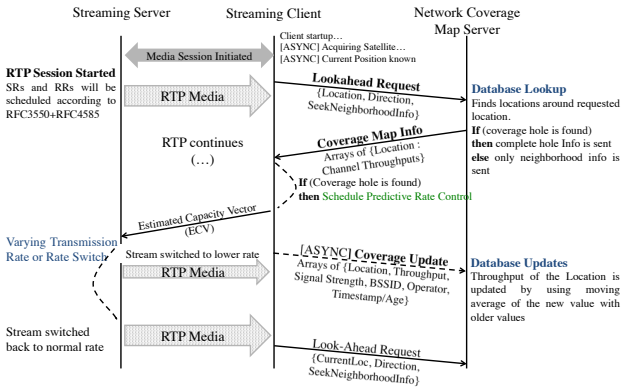


Figure 3: Shows the signaling required to successfully pre-buffer content using the GLASS architecture

Coverage Map Update: The streaming client measures the receiver rate at each geo-location and routinely sends this information to the NCMS. The rate of the updates depends on the speed of the user and the duration of the measurement.

Lookahead Request: The streaming client routinely requests or subscribes to notifications for areas with poor connectivity based on its current trajectory, route (if known) and maximum session media bit rate. The streaming client sends a GET message with its current GPS information (current location, speed, trajectory, etc.) to the NCMS. The NCMS responds with a *Coverage Map Info*.

Coverage Map Info: If the user's travel route is known, then the NCMS returns precise information along that path, else it returns the location, bandwidth information for the locations near the user. In both cases, the NCMS takes the user's speed into account when providing (location, bandwidth) information pairs. For example, the area and granularity of information will be different for a user who is walking or driving.

Rate-control: The streaming client needs to inform the streaming server about the channel capacity so that the server can transmit media at a higher rate to enable pre-buffering. If the streaming server is *GLASS-aware*, the client can provide it with the complete channel conditions, for which the client converts the coverage map information (array of (location, bandwidth) pairs) into temporal information by using the user's vehicular speed and distance between future locations. The temporal representation of the coverage map information is called *Estimated Capacity Vector (ECV)*). Alternatively, if the streaming server is not GLASS-aware but supports RTSP/2.0 [27], the streaming client can use the RTSP Speed: header to control the media transmission rate instead of signaling the complete channel conditions.

IV. Lookahead and Scheduling

The streaming client fetches the coverage map information of the upcoming locations from the NCMS to determine if there are any coverage holes along the expected traveling path (See Figure 4). When the streaming client discovers a coverage hole along its path, it has two options:

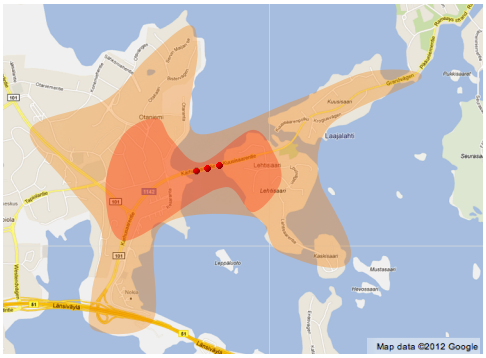


Figure 4: Impact of a coverage hole (the red points on the bridge mark the coverage hole). The streaming client should start buffering just before the edge of the inner territory to be able to playback without any pauses in the coverage hole.

- Pre-fetching and buffering the content is the preferred operation as it would allow the client to maintain the present media quality. The streaming client estimates if it can increase the transmission rate from the server so that it could receive and buffer enough content before

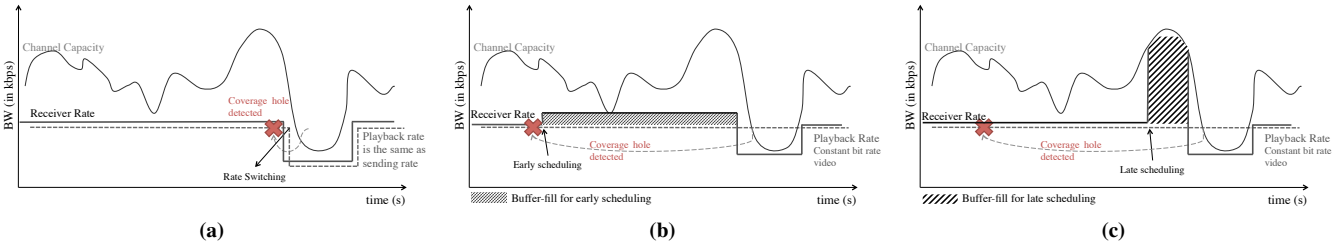


Figure 5: shows the methods of adaptive streaming a) rate-switching, b) early scheduling of buffering, and c) late scheduling of buffering.

reaching the coverage hole (i.e., for T_{tto} seconds) to continue uninterrupted playout throughout the coverage hole. This calculation accounts for the expected capacity along its path to the hole based upon the information received from the NCMS (and its local observations).

- Switching to a lower encoding rate is the fallback if the expected network capacity does not allow raising the transmission rate. This may be peered with pre-buffering: the streaming client may determine the maximum sustainable rate for the upcoming coverage hole and then adjust the transmission rate from the server so that this target can be maintained.

Irrespective of coverage holes, the the streaming client also appropriately schedules the increase and decrease of the transmission rate or rate-switching depending on predicted and observed instant capacity variations; but this type of rate control is not the focus of this paper. In the following subsections we discuss possible solutions for lookahead and scheduling to achieve the above goals.

A. Methods of Lookahead

Known travel route: If the vehicle is using a turn-by-turn navigation system then the streaming client may be aware of the route the user is going take and pre-fetch the media content based on the available bandwidth (from the NCMS) along the route. Therefore, the client can identify the coverage holes at the beginning of the journey and by taking the vehicular speed into account, the streaming client can schedule points at which the media transmission rate can be increased (or switched to a lower encoding rate) and reset. Moreover, the streaming client can routinely query the NCMS for any changes in bandwidth along the travel route.

Area lookahead: Typically, the travel route is not known to the streaming client and in this case it should explore the area around it for coverage holes. In this case, the streaming client sends a *'lookahead request'* to the NCMS with a location, radius and average vehicular speed. The location can be the user's current location or a future location and the radius is the area around that location. Using this mechanism the client routinely queries the NCMS for upcoming locations for detecting coverage.

The NCMS responds to the *'lookahead request'* with the *'Coverage Map Info'*. NCMS samples the neighborhood locations based on the vehicular speed and creates the throughput vector for the locations in that area. If a coverage hole is detected, the NCMS provides the additional information even if it exceeds the requested radius.

Coverage hole subscription: After the streaming client has obtained the *'Coverage Map Info'* for its upcoming path, it will not query this area again on its own unless its path changes. Yet, the NCMS server may have updates about changes along the predicted path and needs to notify the streaming client about those. Therefore, any path query by a client is treated as an implicit subscription for updates (using long poll or can be implemented using *websockets*). The subscription is valid until the streaming client passes by that point or if the route changes (or by timeout).

B. Moment of Detection

When to increase the transmission rate or switch to a lower rate depends on how much in advance a coverage hole is detected.

Late Detection and Rate-switching: The streaming client detects the coverage hole a few seconds before entering it. The client in this case has just enough time to switch the media rate to a lower encoding rate. To make sure that there is no disruption the NCMS provides the lowest and the average bandwidth in the coverage hole. Figure 5(a) shows rate-switching due to late detection.

Early Detection and Early Scheduling: The streaming client detects the upcoming coverage hole well in advance; i.e., there is sufficient time and available bandwidth to increase the transmission rate of the media stream to compensate the reduction in the coverage hole.

The client may then increase the receiver buffer size to accommodate more media packets. The drawback of this scheme is that if the client starts to buffer very early and the user ends the playback or does not pass through the coverage hole (because of changing travel routes) then the media data is unnecessarily buffered. Figure 5(b) shows the early detection and slight increase in transmission rate to overcome the coverage hole.

Early Detection but Late Scheduling: The streaming client detects the upcoming coverage hole well in advance

but the client schedules the increase of the transmission rate such that the extra available bandwidth over the scheduled period (T_{tto}) is able to compensate for the missing bandwidth in the coverage hole. Inside the coverage hole, the streaming client reduces the transmission rate to the average or minimum rate of the coverage hole.

Figure 5(c) shows the late scheduling and the increase in buffering. The size of the extra buffer in early and late scheduling is the same; however, late scheduling reduces the chance of unnecessary buffering due to travel path changes, but relies more on the prediction accuracy.

V. Protocol Design and Implementation

A. Protocol

The streaming client interacts with the NCMS over HTTP and with the streaming server using RTCP extensions. We use HTTP, RESTful APIs and JSON due to their convenience in development. However, these messages can be encoded in binary 32-bit word format (like TCP's header format) to reduce signaling overhead.

Coverage Map Update: The streaming client sends a POST message with geolocation (latitude, longitude), speed, average receiver rate and loss rate at that location or in the past five seconds, and time when the updated was created (utime). The streaming client should also include the authentication hash, which it received during login. NCMS will respond with either a *'No Error'*, *'Missing Parameter'*, *'Invalid Hash'* or *'Unable to update'*. In the case of lack of connectivity the streaming client holds on to the update messages and sends it later as a JSON dictionary when the connectivity is restored.

Sample Single Coverage Map Update:

```
https://<servername>/backend/update.json?id=<hash>
&latlon=60.16865,24.82348&s=7kmph&rr=1300&lossrate=0.0
&m=drive&utime=1314952356&op=Elisa&typ=3G&probe=false
```

Sample JSON Response:

```
{"err":"No error"}
```

We assume that the NCMS is able to group or cluster nearby geographical points that have similar throughput. We also assume that the NCMS is able to split existing clusters into smaller groups if points with lower or higher throughput appear in the same geographical cluster. The design and implementation of the NCMS is outside the of scope of this paper.

Lookahead Request and Coverage Map Info: The streaming client routinely queries the NCMS for future locations. It is similar to how a user explores a map in multiplayer games (also called *fog of war*). The streaming client issues one or more *lookahead* requests, each lookahead request is sent as an HTTP GET request and contains location, expected media rate, vehicular speed and a range around the requested location.

The NCMS uses the location as a central point and fetches points from the coverage database within the requested range. The NCMS may additionally reduce the set of locations to a smaller set. For example, if the user is moving at a higher speed, it may filter-out locations with a small impact range. If the NCMS finds a coverage hole (an area where throughput is less than the requested media rate), it also includes the area around the coverage hole, even if it exceeds the requested range.

The NCMS responds to the lookahead request with the *'Coverage Map Info'* which is a JSON dictionary of points with locations and their radius, throughput and the NCMS confidence in the measurement. It also flags the report if it found a coverage hole.

Sample Lookahead Request:

```
https://<servername>/backend/lookahead.json?
id=<hash>&latlon=60.16444,24.92099&range=500m
&m=drive&s=7kmph&mrate=1000kbps
```

Sample Coverage Map Info (JSON encoding):

```
{"user":7, "coverage_flag":0, "Coverage Map Info":[
 {"latlon":"60.16474,24.9121", "radius":"35m", "mode":
   "drive", "throughput":"1375 kbps", "confidence":"low"},
 {"latlon":"60.16445,24.91591", "radius":"95m", "mode":
   "drive", "throughput":"1290 kbps", "confidence":"high"},
 {...}], "err":"No error"}
```

On finding the coverage holes, the streaming client uses Google Map's Direction API [28] for calculating the route to the coverage hole. The direction API provides the total length, duration to arrive at the location, and the individual *'legs'* of the route. If the Direction API is unavailable then the client approximates the route as the straight line (geodesic) between current location and the location of the coverage hole. The points in the lookahead that lie approximately along the straight path form the waypoint markers to the coverage hole.

Sample Direction Request:

```
https://maps.googleapis.com/maps/api/directions/json?
origin=60.16444,24.92099&destination=60.16445,24.91591
&sensor=true&mode=walking
```

If the streaming client detects that it is on the route to the coverage hole, the client schedules the pre-buffering using Algorithm 1. The client sends the Estimated Capacity Vector (ECV) for adjusting the transmission rate or Temporal Maximum Media Stream Bit rate Request (TMMBR) [RFC5104] for switching to a different encoding rate.

B. When to schedule the ECV?

First, the streaming client matches the locations received in the Direction API to the (location, range, throughput) tuples received in the *Coverage Map Info*. Second, the streaming client calculates the minimum time required to pre-buffer the media based on the travel route. The streaming client does this by calculating the time required to travel between each waypoint ($\frac{distance}{speed_{vehicular}}$) and multiplying it by the throughput between those points. The transmission rate is throttled at the point where the calculated pre-buffer

Algorithm 1 Scheduling the ECV or TMMBR at the streaming client

Require: Streaming knows:
1. Location and range of Coverage hole.
2. Waypoints: Route from current location to coverage hole.

Ensure: Streaming client has the throughput data for all waypoints. Else issue lookaheads for those legs of the route.

{Merge the waypoints from the route and the NCMS to create a consistent throughput map of points. For legs or locations with no bandwidth information assume the throughput in the vicinity as an approximation. If the approximation is incorrect, it will not help the current user but the next user who travels the same route.}
$T_{gap} = \frac{2 \times radius_{hole}}{speed_{vehicle}}$
$Buffer_{extra} = \frac{(rate_{media} - thoughput_{hole})}{8} \times T_{gap}$ KB
$N = num\_of\_points(route)$
$X \leftarrow -1$, offset $\leftarrow 0, ecv_t[N-1] \leftarrow T_{gap}$
for $i = N-2$ to 0 **do**
  {we are traversing the route backwards from a location (N-2) before the coverage hole (N-1, N) to the current location (0)}
  $ecv_t[i] \leftarrow \frac{geodist(route[i].location, route[i+1].location)}{speed_{vehicle}}$
  {for geodesic distance calculation, we use Haversine Formula}
  $ecv_b[i] \leftarrow \frac{route[i].throughput - rate_{media}}{8} \times ecv_t[i]$
  $b \leftarrow b + ecv_b[i]$
  if $X \leq -1$ **and** $b > Buffer_{extra}$ **then**
    $X \leftarrow i$
  else
    if $X > -1$ **then**
      offset $\leftarrow$ offset $+ ecv_t[i]$
    end if
  end if
end for
if $X \neq -1$ **then**
  $j \leftarrow 0$.
  for $i = X$ to $N-1$ **do**
    {N because we want to reduce the transmission rate during the coverage hole or completely pause it. N to N-1 is the diameter of the coverage hole}
    $ecv[j].time \leftarrow offset + ecv_t[i]$
    $ecv[j].throughput \leftarrow route[i].throughput$
    $j \leftarrow j + 1$
  end for
else
  send TMMBR before the edge of the hole or try to recursively reset old scheduling points.
end if

is sufficient to compensate for the deficit of the coverage hole.

If the path does not allow sufficient pre-buffer then a TMMBR is scheduled near the edge of the coverage hole. If the average vehicular speed of the user changes dramatically (e.g. the user goes from walking to public transport), then the ECV needs to be recalculated, mere scaling ($\delta_{scale} = \frac{speed_{old}}{speed_{new}}$) will not work because the time duration of the throttling may be insufficient to pre-buffer the deficit from the coverage hole.

If another coverage hole is detected then the streaming client calculates if there is sufficient time between the holes to pre-buffer content. If it is possible to pre-buffer content between the two coverage holes then the streaming client sends the ECV for the coverage hole after passing through the first coverage hole. If the time to pre-buffer is insufficient then the streaming client should calculate if there is enough time between now and the scheduling of the first ECV to pre-buffer for the new coverage hole. This is done recursively starting from the newest coverage hole to oldest and using the Direction API between each coverage hole. If it is possible then an updated ECV is sent to the streaming server covering all the congestion spots. If it is not possible to reschedule the previous ECVs then the steaming client switches to lower encoding rates (using TMMBR) for the coverage holes that it is unable to pre-buffer content.

C. Implementation

We developed an adaptive multimedia streaming system using open-source libraries, including Gstreamer[3], x264[4] and JRTPLIB[5]. It can encode and decode files, so that we can use it both as streaming server and client.

Whenever the location changes, a system service provides the streaming client with GPS updates. The streaming client measures its receiver rate every few seconds or on location change and sends updates to the NCMS. If the coverage is limited, the streaming client stores these updates in a file and sends them when the connectivity is better.

We built a minimalistic NCMS using a web-framework and PostgreSQL[6]. NCMS and the media server run on a single Linux server (Ubuntu 10.10). For our evaluation, NCMS uses a K-means [25] and K-NN [26] clustering algorithms to group nearby points with similar bandwidth ($\pm 25\%$) and the NCMS reports the average values of the latest group.[7] The streaming server uses the "Football" video sequence pre-encoded in five media rates (64, 128, 256, 512, 768 and 1000 kbps) using the x264 encoder. All the media files are 265 seconds long, 30 FPS and a GOP=16. We use a small group of pictures (GOP) so that the stream can be switched at 1 second timescales.

D. Micro-benchmarks

To test the system before using it for real-world experiments, we simulate the 3G link variation using the RLC traces defined in [29]. We assign locations along a path with the throughput from the traces. To add some variation between each successive run, we multiply the channel capacity at those locations with a random value, uniformly distributed in $[0.5; 1.5]$. For these experiments, we also implement a *link simulation module* that simulates a user moving along a geographical path and reads the associated throughput and loss rate from the RLC file. The *link simulation module* uses Dummynet [30] to control the link characteristics and shares only its current location with the streaming client to communicate with the NCMS. To clarify, only the travel

[3]http://gstreamer.freedesktop.org/
[4]http://www.videolan.org/developers/x264.html
[5]http://research.edm.uhasselt.be/~jori/jrtplib/documentation/
[6]http://www.postgresql.org/
[7]The NCMS and its parameters are outside the scope of the paper.

path and the 3G link characteristics are simulated while the rest of the system (NCMS, streaming client and server) is the same as the one deployed.

We evaluate the performance of the following schemes:

1) **No adaptation:** The streaming client sends normal RTCP receiver reports but the streaming server neither adapts nor performs any rate-switching. In this case, bit-error or congestion-induced packet loss will cause buffer under-runs and lead to poor video quality and low user experience.
2) **Omniscient adaptation:** The streaming server is aware of the exact end-to-end channel capacity and performs the perfect rate-adaptation.
3) **Rate-Switching using GLASS:** The streaming client performs short lookahead request and is only able to detect coverage holes moments before the user arrives at the location. In this case, the streaming client sends a TMMBR message (with lowest coverage hole bit rate) before entering the coverage hole and another one after the coverage hole (to reset the bit rate). The streaming server reacts by switching the media stream to the closest available media rate.
4) **Late scheduling using GLASS:** The streaming client performs lookahead requests and tries to detect coverage holes much before the user arrives at those locations. To reduce the impact of changing routes, the streaming client uses the late scheduling proposed in Algorithm 1.

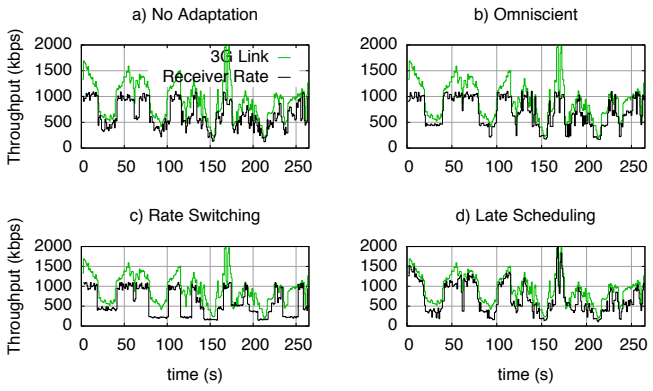


Figure 6: Comparison of the receiver rate to the 3G link rate (based on RLC traces [29]. a) receiver rate when the streaming server provides no adaptation, b) rate-switching when the streaming server is fully aware of the network characteristics and is therefore able to adapt the sending rate perfectly, c) rate-switching when the streaming-client relies on short lookaheads, and d) pre-emptive buffering to overcome the coverage holes. In all the cases, the streaming client initiates the session with a 1Mbps media stream.

We use the following metrics to compare the above schemes: Packet Loss Rate (PLR), average receiving rate, average Peak Signal-to-Noise Ratio (PSNR of the Y-component). Figure 6 compares the receiver rate using each mechanism (in black) to the 3G link throughput (in green). In Figure 6(a) the streaming client does not adapt to the changing channel capacity and produces a low quality of experience ($PSNR_{avg} = 17$) while (b) produces the best possible using rate-switching ($PSNR_{avg} = 41.1$). Using recommendations from the NCMS, the streaming client performs comparable to the omniscient scenario in terms of PSNR; rate-switching ($PSNR_{avg} = 39.3$) or pre-buffering ($PSNR_{avg} = 39.1$). However, the average media rates are widely different (See Table I).

Table I: Comparison of performance metrics of adaptation mechanism over 10 independent runs

Method	RR_{avg}	$PSNR_{avg}$	σ_{PSNR}	PLR
No adaptation	645.4	19.48	2.55	17.02
Omniscient	700.2	41.1	0.49	0.0
Rate-switching	579.4	39.3	1.89	0.0
Late scheduling	713.1	39.1	1.57	0.0

VI. Real-World Traces

We collect throughput traces of the Helsinki metropolitan area by traveling on the regional transport service (Buses, Trams) and by walking around the Helsinki city-center and the university area (*Otaniemi*). The media server streams a constant bit rate video of *2Mbps* to a Nokia N900. The N900 receives the stream over its 3G interface and uses the internal GPS module for location tracking. The 3G provider advertises connectivity up to 2Mbps. The streaming clients send coverage map updates to the NCMS so that the NCMS is able to monitor the available throughput in these regions. In total, the NCMS collected over 400,000 updates over a month of operation (about 40-50 bus-trips). For 6 geographical areas, the NCMS received more than 10,000 updates (*Otaniemi*, Helsinki City Center) while on average each geographical had about 10 to 100 updates. Figure 7 shows the average throughput per location in an around the university (*Otaniemi*). This map can be interpreted as a vector the streaming client would receive on performing a lookahead.

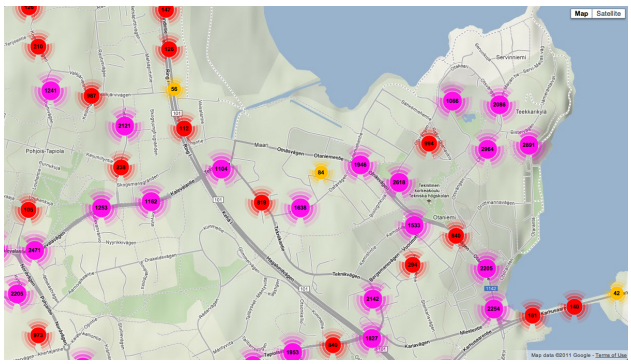


Figure 7: Throughput traces around Otaniemi (University Area).

VII. Performance Evaluation

After the NCMS has gathered the throughput traces, we conduct simulated experiments of users traveling at different vehicular speeds through the Helsinki region. The simulation setup is exactly the same as the one used for the *micro-benchmarks*, but instead, we use the actual traces from the individual trips around the city to simulate the scenarios. Each simulation is run 10 times and measures the Average PSNR, loss rate and sending rate. The channel characteristics are roughly the same for each run, but the *coverage map information* reported by the NCMS may change because the streaming clients report their coverage along the route during each run.

We evaluate the performance of the information received in the lookahead requests from the NCMS. We consider two scenarios, traces with low variability in vehicular speed (*driving down the highway, walking*) and traces with high variability in vehicular speed (*driving in stop-go-stop-type of traffic*). The real-world traces (Figure 8) reveal that throughput can vary dramatically over several days.

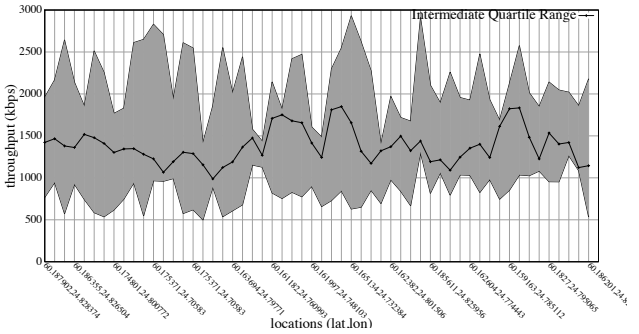


Figure 8: Example of the maximum, minimum and average of the Inter Quartile Range ($IQR = \pm 1\sigma$) throughput along a route over several days. The average number of data points per location is 475. Note: the traces are not gathered all the time at the locations but only when passing through the area. The throughput may bear correlation to time of day but we have insufficient information to make the correlation.

A. Low Variability in Connectivity

In this case, the user's vehicular speed is kept constant and the simulator moves the user smoothly through the locations on the route. Consequently, the streaming client is more confident about the information it receives in the lookahead and does not need to query the NCMS often for updates.

In this scenario, there are intermittent gaps in coverage but overall the throughput at most locations is above the required media rate. The omniscient scenario provides the best rate-adaptation but since it is unable pre-buffer, the PSNR ($PSNR_{avg} = 43$) is lower than that of late scheduling ($PSNR_{avg} = 48$). Alternatively, performing no adaptation causes frequent re-buffering and packet discards which is reflected in the low PSNR ($PSNR_{avg} = 27$). Rate-switching using small lookahead requests has comparable PSNR ($PSNR_{avg} = 42.75$) to Omniscient ($PSNR_{avg} = 43.12$) and has no packet losses. Rate-switching avoids all drops by switching to values lower than the average or minimum reported throughput. Late scheduling outperforms the rest of the schemes in terms of PSNR ($PSNR_{avg} = 48.43$) because it predictively pre-buffers content and does not have to rate-switch. This is largely due to the low density of coverage holes and good connectivity between them.

Table II: Bus ride with good 3G coverage

Method	SR_{avg}	$PSNR_{avg}$	σ_{PSNR}	PLR
No adaptation	865	27.48	4.55	6.6
Omniscient	929	43.12	1.9	0.33
Rate-switching	881	42.75	2.21	0.0
Late scheduling	1014	48.43	0.18	0.0

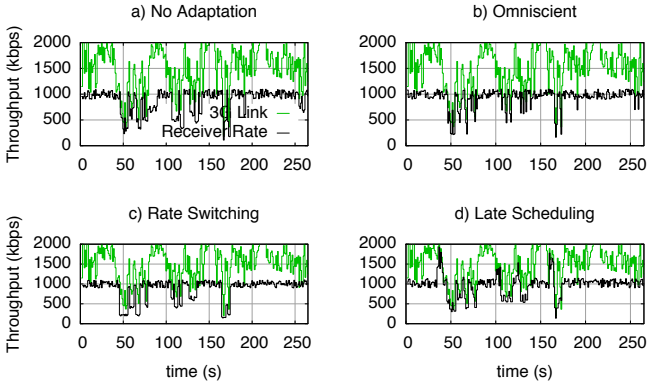


Figure 9: Variation of receiver rate to 3G link capacity based on a) no adaptation b) Omniscient c) Rate-switching d) Late scheduling when there are few coverage holes and good connectivity between them.

Table II compares the results for the different schemes and Figure 9 shows the sending/receiver rate variation to channel throughput for one out of 10 runs. Late scheduling shows an increase in sending rate before arriving at a coverage hole, which shows that the streaming client pre-buffers content that it would normally lose during the coverage hole. Whenever the sending rate is much lower or far above the throughput (figures 9(c) and (d)) this is because the throughput reported by NCMS has not converged to the current values.

B. High Variability in Connectivity

In this case, the user's vehicular speed is variable between locations and it frequently stops and moves. Consequently, the streaming client is not very confident about the information it receives in the lookahead and sometimes re-queries the NCMS for updates. Due to the variability in the user's movements, both omniscient and late scheduling mechanisms are unable to avoid losses (0.21% and 0.19%, respectively). Rate-switching is again able to avoid all losses by choosing rate-switching to values lower than the minimum throughput in the coverage hole and therefore has lower average PSNR ($PSNR_{avg} = 43$). While late scheduling

($PSNR_{avg} = 45$) and Omniscient ($PSNR_{avg} = 46$) have higher PSNR, both the mechanisms can use NACKs to fetch missing packets (*the NACKs are sent just once and only for I-frames*) and the NACKs also contribute to the higher sending rate. The variability in user's motion also causes the streaming client to use an additional $9kbps$ for re-querying the NCMS and sending the updated ECVs to the streaming server.

Table III: Bus ride with poor or variable 3G coverage

Method	SR_{avg}	$PSNR_{avg}$	σ_{PSNR}	PLR
No adaptation	948.6	23.12	5.2	5.3
Omniscient	970.9	45.23	0.58	0.21
Rate-switching	901.8	43.46	0.46	0.0
Late scheduling	1047.1	46.37	1.707	0.19

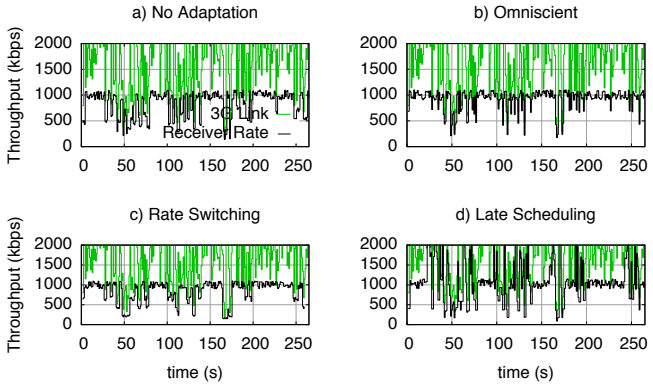


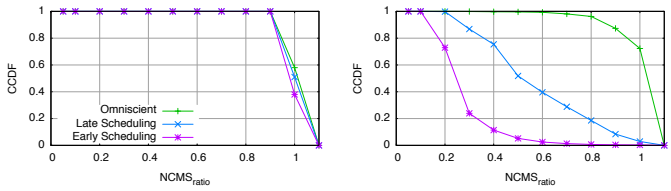
Figure 10: Variation of receiver rate to 3G link capacity based on a) no adaptation b) Omniscient c) Rate-switching d) Late scheduling when there is variability in user's motion mainly due to *stop-go-stop* type of vehicular traffic.

Table III compares the results for the different schemes and Figure 10 shows the sending/receiver rate variation to channel throughput for one out of 10 runs. In Figure 10, late scheduling shows variability in sending rate this is because the throughput reported by NCMS to the streaming client are updated by re-queries or the NCMS has values different from the current throughput.

To summarize, the streaming client may prefer pre-buffering using late scheduling when the lookahead requests reveal rather stable network conditions for throughputs greater than the requested media bit rate. Consequently, for locations with frequent throughput variations the streaming client may prefer to perform rate-switching. If the NCMS reported values in the lookahead are different from the current conditions observed by the streaming client then the streaming client may choose to ignore the information in the lookahead requests and perform reactive rate-control.

C. Comparison between Early and Late Scheduling

$NCMS_{ratio}$ is the ratio of the NCMS recommended throughput and the current throughput. A value greater that one indicates that the NCMS recommended value is higher than the actual and the streaming client may experience losses due to over-utilization. While a value lower than one indicates that the NCMS will under-utilize the link and would require more time to pre-buffer content than actually is needed. The best performance would only be possible if the actual end-to-end throughput is reported by the streaming clients and that the available throughput does not change in the interim.



(a) after t=1h **(b)** after t=8h

Figure 11: The distribution of locations based on $NCMS_{ratio} = \frac{NCMS\ recommended\ throughput}{ActualThroughput}$ after a) one hour, b) after 8 hours. The plot shows that the NCMS reports lower throughput when the streaming client's use Early Scheduling compared to Late Scheduling.

Figure 11 shows that the NCMS *under-utilizes* the available throughput in many area when using early scheduling because the streaming clients schedule the increase in sending rate much too early (upon detection) and are therefore not exploring the available throughput. Alternatively, in late scheduling the streaming client requests the increase in sending rate closer to the coverage hole, thus exploring the available throughput at the edges of the coverage hole. Therefore, in the case of late scheduling, the NCMS nearly always has a good $NCMS_{ratio}$ near the coverage hole and poor $NCMS_{ratio}$ at other places where the available throughput is higher than the maximum bit rate (1Mbps).

To summarize, the throughput information received from an NCMS lookahead should be used as a recommendation (indicator) and not as a directive by the streaming client. If possible, the streaming client should report not only the received media rate (by the multimedia flow) but also monitor system level throughput.

VIII. Conclusions

We presented a mechanism enabling a media streaming client to proactively react to upcoming capacity limitations in wireless access networks. The streaming client uses hints provided by a network coverage map service (which could apply crowd sourcing to gather the respective coverage and capacity information). These hints are sent to the streaming server so that the client can pre-buffer and, if needed, switch to a different media rate.

We find that the information provided by the NCMS is suitable for both predictive rate-switching and pre-buffering and helped in avoiding almost all packet losses in the

scenarios we investigated, noticeably increasing video quality. While these results are promising, they are only a first step: flash crowd effects resulting from many clients approaching and/or creating a coverage hole at the same time require further study. Similarly, the performance impact for a more encompassing geographical coverage and more diverse network connectivity deserves further investigation.

Beyond this, two areas appear particularly interesting: combining these prediction mechanisms with RTP-based congestion control and applying this very mechanism to HTTP-based (instead of RTP) streaming. Finally, a scalable and trustworthy crowd-sourcing architecture, e.g., implemented as an operator service, is an obvious prerequisite for real-world deployments.

REFERENCES

[1] Google Code, "YouTube Data API." [Online]. Available: http://code.google.com/apis/youtube/2.0/reference.html

[2] Allot MobileTrends, "Global Mobile Broadband Traffic Report," Allot Communications, Tech Report, July 2011.

[3] M. Zink, O. Künzel, J. Schmitt, and R. Steinmetz, "Subjective Impression of Variations in Layer Encoded Videos," in *Proc. of IWQoS*, 2003.

[4] 3GPP TS 23.107, "Quality of Service (QoS) concept and architecture."

[5] I. Curcio, V. Vadakital, and M. Hannuksela, "Geo-predictive Real-time Media Delivery in Mobile Environment," in *Proc. of Mobile video delivery*, oct 2010.

[6] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Bitrate and video quality planning for mobile streaming scenarios using a GPS-based bandwidth lookup service," in *Proc. of ICME*, july 2011.

[7] R. Rejaie, M. Handley, and D. Estrin, "RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the Internet," in *Proc. of INFOCOM*, mar 1999.

[8] S. Cen, C. Pu, and J. Walpole, "Flow and congestion control for internet media streaming applications," in *Proc. of Multimedia Computing and Networking*, 1998.

[9] H. Kanakia, P. Mishra, and A. Reibman, "An adaptive congestion control scheme for real-time packet video transport," *ACM SIGCOMM CCR*, vol. 23, no. 4, 1993.

[10] A. Diaz, P. Merino, L. Panizo, and A. Recio, "Evaluating video streaming over GPRS/UMTS networks: A practical case," in *Proc. of IEEE VTC*, 2007.

[11] 3GPP TS 26.234, "Transparent end-to-end Packet-switched Streaming Service (PSS); Protocols and codecs," 2007.

[12] 3GPP TS 26.346, "Multimedia Broadcast/Multicast Service (MBMS); Protocols and codecs," 2007.

[13] T. Schierl, Y. Sanchez de la Fuente, R. Globisch, C. Hellge, and T. Wiegand, "Priority-based Media Delivery using SVC with RTP and HTTP streaming," *in Proc. of Multimedia Tools and Applications*, 2010.

[14] C. Krasic, J. Walpole, and W. Feng, "Quality-adaptive Media Streaming by Priority Drop," in *Proc. of ACM NOSSDAV*, 2003.

[15] C. D. S. Akhshabi, A. Begen, "An Experimental Evaluation of Rate-Adaptation Algorithms in Adaptive Streaming over HTTP," in *Proc. of ACM MMSys*, Jan 2011.

[16] L. D. Cicco and S. Mascolo, "An Experimental Investigation of the Akamai Adaptive Video Streaming," *HCI in Work and Learning*, Jan 2010.

[17] R. Kuschnig, I. Kofler, and H. Hellwagner, "An Evaluation of TCP-based Rate-Control Algorithms for Adaptive Internet Streaming of H.264/SVC," in *Proc. of ACM MMSys*, 2010.

[18] E. Brosh, S. A. Baset, D. Rubenstein, and H. Schulzrinne, "The Delay-Friendliness of TCP," in *Proc. of ACM SIGMETRICS*, 2008.

[19] D. Kutscher and J. Ott, "Service Maps for Heterogeneous Network Environments," in *Proc. of MDM.*, May 2006.

[20] P. Aravinda, K. Acharya, A. Sharma, E. Belding, K. Almeroth, and K. Papagiannaki, "Congestion-Aware Rate Adaptation in Wireless Networks: A Measurement-Driven Approach," in *Proc. of IEEE SECON*, 2008.

[21] P. Shankar, T. Nadeem, J. Rosca, and L. Iftode, "CARS: Context-Aware Rate Selection for Vehicular Networks," in *Proc. of IEEE ICNP*, 2008.

[22] J. Yao, S. Kanhere, and M. Hassan, "An empirical study of bandwidth predictability in mobile computing," in *Proc. of ACM WiNTECH*, 2008.

[23] ——, "Geo-intelligent Traffic Scheduling For Multi-Homed On-Board Networks," in *Proc. of ACM MobiArch*, 2009.

[24] ——, "Quality improvement of mobile video using geo-intelligent rate adaptation," in *Proc. of IEEE WCNC*, 2010.

[25] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "A local search approximation algorithm for k-means clustering," in *Proc. of SCG*, 2002.

[26] G. S. Iwerks, H. Samet, and K. Smith, "Continuous K-nearest neighbor queries for continuously moving points with updates," in *Proc. of VLDB*, 2003.

[27] Schulzrinne, H., Rao, A., Lanphier, R., Westerlund, M., and M. Stiemerling, "RTSP 2.0," 2011, (Work in progress). [Online]. Available: draft-ietf-mmusic-rfc2326bis

[28] Google Code, "Google Maps API." [Online]. Available: http://code.google.com/apis/maps/documentation/webservices/index.html

[29] 3GPP R1-081955, "LTE Link Level Throughput Data for SA4 Evaluation Framework." May 2008.

[30] M. Carbone and L. Rizzo, "Dummynet revisited," *ACM SIGCOMM CCR*, Jan 2010.