On the Use of Location Window in Geo-Intelligent HTTP Adaptive Video Streaming

Jannatul Fardous
School of Computer Science and Engineering
University of New South Wales, Sydney
NSW-2052, Australia
Email: jannatulf@cse.unsw.edu.au

Salil S. Kanhere
School of Computer Science and Engineering
University of New South Wales, Sydney
NSW-2052, Australia
Email: salilk@cse.unsw.edu.au

***Abstract*—HTTP adaptive video streaming has become the de facto standard for media data delivery in the Internet. Mobile users are increasingly accessing video streaming services while traveling in fast-moving vehicles (e.g., public transport). The inherent high-speed mobility in these scenarios escalates bandwidth uncertainty and seriously degrades the performance of HTTP adaptive video streaming. This paper proposes a *location window* based *geo-intelligent* adaptive streaming algorithm, which adapts to the geo-spatial bandwidth variations experienced by a fast-moving user by adjusting the quality of the next chunk based on the estimated bandwidth at the next *X* locations of the mobile user. In order to realize geo-intelligence, we introduce a neural network model for accurately creating bandwidth maps that store location-specific bandwidth knowledge. By incorporating both these contributions in conjunction with real-world mobile broadband bandwidth traces from a metropolitan area, we present a systematic study to explore the effects of varying the size of the location window on the user-perceived Quality of Experience (QoE). The evaluation results demonstrate that an optimum *location window* can be identified, which can almost entirely eliminate playout buffer underruns, thus leading to a smooth and high-quality streaming experience.**

I. Introduction

Given the widespread coverage of high-data rate mobile networks (e.g., HSDPA and LTE) and the emergence of personal mobile devices with high resolution displays and fast processing speeds (e.g., smartphones and tablets), video streaming is now one of the fastest growing mobile applications. In particular, commuters during their daily travels are increasingly viewing videos on popular websites such as Youtube and Vimeo. While in the past most video streaming technologies utilized streaming protocols such as RTP, current streaming services are almost exclusively based on adaptive HTTP streaming. The source video is encoded at multiple bitrates (i.e. different qualities) and segmented into chunks. The core idea involves dynamically switching the bitrate of the chunks according to the available network bandwidth.

The capability of the application to accurately estimate network bandwidth is a key element of such adaptations. In the absence of user mobility, however, bandwidth variations are not as frequent or severe. Consequently, relatively simple techniques can be employed to estimate network bandwidth. For example, in current implementations of adaptive HTTP streaming, the servers periodically communicate with the clients to probe the current bandwidth and buffer occupancy [1] and then use the client feedback to make the adaptation decisions. High-speed mobility escalates bandwidth uncertainty for the mobile users. While travelling at vehicular speed, a mobile user changes his geographic location at a fast rate. Since wireless link quality is sensitive to location, rapid location change introduces frequent variation in link quality, which ultimately results increased instability for the mobile bandwidth. For example, a recent field study found that, while traveling in public transport, the mobile bandwidth can vary from 0 kbps to 3500 kbps [2]. As mobile broadband capacity (i.e. peak data rate) continues to increase, the bandwidth variations will become even more significant, because larger capacity means a wider range for the bandwidth to fluctuate within. A greater variability in network bandwidth means more serious impact on application performance should the application fail to adapt to these variations. Recent research [3], [4] demonstrates that the reactive techniques employed in HTTP adaptive streaming fail to capture the bandwidth dynamics of a mobile client, leading to ineffective adaptation and poor application performance. At the client end, this is manifested as frequent pauses in playback caused by the exhaustion of the playout buffer (also referred to as buffer underrun).

Since (change of) location is a key factor in the escalation of bandwidth uncertainty in high-speed mobility, intuitively it makes sense to gather as much location-specific knowledge as possible. This *a priori* location-specific knowledge can be used to better predict bandwidth variations and hence achieve better application adaptation in a moving scenario. This idea was first proposed by researchers in [5], wherein, the term *geo-intelligence* was coined to refer to this form of location-specific bandwidth knowledge in mobile computing. Past location-specific bandwidth performance data is stored in *bandwidth maps*, by mapping the average value of historical network bandwidth observations to the road network. These maps provide information such as ”if you are the intersection of Streets X and Y, you can expect average bandwidth of Z Mbps from provider A”. The authors proposed a simple geo-intelligent adaptive streaming algorithm, which changes the bitrate of the stream according to the estimated bandwidth of the next location of the mobile user obtained from the bandwidth map. In [6], the authors present another instantiation of geo-intelligence, wherein, the estimated bandwidth at all the remaining locations along the route is used to determine the streaming quality for the next chunk. These heuristics represent two extreme endpoints (one location vs all remaining locations) in the design space of *look-ahead* strategies that adapt the streaming bitrate based on the estimated bandwidth at the upcoming *X* locations for the mobile user, which we refer to as the *location window*. Intuitively, considering a small value for the location window, may not be sufficient for dealing with sharp fluctuations in bandwidth (e.g., sudden outage) that may arise at latter locations. On the other hand, if the window is very large, then the impact of potential errors in bandwidth estimation is compounded, which can lead to incorrect bitrate selection decisions, leading to sub-optimal streaming performance. A systematic study that examines the impact of varying the size of the location window in geo-intelligent HTTP adaptive streaming on the user-perceived QoE has not yet been undertaken, which is precisely the goal of this paper.

Specifically, this paper makes the following contributions:

- We propose a window-based geo-intelligent adaptive HTTP streaming algorithm, which adapts the streaming quality of the next chunk based on the estimated bandwidth along the next *X* locations of the mobile user.
- Another important consideration for realizing geo-intelligence is accurate estimation of the location-specific bandwidth based on historical data. Past research has used simplistic averaging methods for this purpose. We propose a neural network model for creating bandwidth maps. We demonstrate using real-world traces collected from a moving vehicle in Sydney, that our model can achieve significantly better accuracy in estimating mobile network bandwidth.
- We undertake extensive evaluations using real mobile bandwidth traces to study the impact of varying the size of the location window, *X*, on the streaming performance. The evaluation results indicate that it is indeed possible to find an optimal location window that can almost entirely eliminate buffer underrun (i.e. freezing occurrence), thus resulting in a smooth and high-quality streaming experience.

The rest of the paper is organised as follows. Section II introduces related work. Section III presents our neural network based approach for creating bandwidth maps. Section IV elaborates the location window based strategy that we have used in our study. Trace-driven evalution results are presented in Section V. Finally, Section VI concludes the paper.

II. Related Work

While HTTP adaptive streaming technique has been successful and widely adopted for media data delivery, it still suffers from sub-optimal performance for a fast moving user. This is reported by recent work in [3], which undertook an empirical evaluation of HTTP adaptive streaming in vehicular mobility and found that a user driving along a typical route in a city, would experience several instances when the playout buffer would empty out, which in turn would cause the video to freeze. To overcome these problems, several strategies have been proposed in literature. An optimization algorithm based on the buffered data and transmission time of the chunk was proposed [4]. In [7], a new rate-adaptation algorithm was proposed, that estimates the current link condition based on the segment fetch time. A client intelligence mechanism using MDP (Markov Decision Process) that selects the quality by estimating the transmission time of the next chunk appears in [8]. All of the aforementioned strategies are reactive in nature and are hence unlikely to overcome the core problem of extremely rapid and frequent bandwidth fluctuations, inherent in high-speed mobility. By the time a reactive link monitor learns about the current state of the network, the mobile user would have already moved to a new location, where the bandwidth conditions are likely to be different. As such, all of these strategies are likely to experience several instances of buffer underrun during a typical usage scenario.

A contrasting proactive approach to solve this problem is based on the paradigm of geo-intelligence, the idea of mapping past mobile bandwidth performance to road networks. The authors in [9], were the first to report using empirical measurements that mobile bandwidth is strongly correlated with the geographical location in a vehicular scenario. They showed that mobile bandwidth is more predictable when a location-specific historical statistics of the bandwidth are maintained. The same authors proposed a UDP-based streaming algorithm that incorporated geo-intelligence with TCP Friendly Rate Control (TFRC), called Geo-TFRC. In this mechanism, the server chooses the quality (i.e. bitrate) corresponding to the bandwidth of the next location of the mobile user. However, since this strategy only looks ahead one location, it is unable to react early enough if there are significant bandwidth fluctuations coming up, and thus ultimately causing buffer underruns in such situations. A similar idea was also proposed in [10], but their primary focus was only on dealing with bandwidth outages. More recently, [6] presents another instantiation of the principle of geo-intelligence in the context of HTTP adaptive streaming. The algorithm chooses the quality by considering the estimated network bandwidth at all remaining locations along the route. Since the look-ahead window is large, the impact of potential errors in bandwidth estimation is compounded, which can lead to incorrect bitrate selection decisions, leading to sub-optimal streaming performance.

The above research describes two extreme endpoints (one location vs all remaining locations) of look-ahead strategies for geo-intelligent video streaming. However, the effects of varying the number of upcoming locations (i.e. *location window*) on the users QoE has not been systematically addressed and examined. In this research, we explore this design space of look-ahead strategies to determine if there is an optimal window size that achieves the best possible performance.

III. A Neural Network Model for Creating Geo-Intelligence

As discussed in Section I, bandwidth maps play an important role in delivering geo-intelligent services. Past location-specific bandwidth performance data is stored in *bandwidth maps*, by mapping statistical properties of the historical network bandwidth observations to the road network. Prior research [5], [6], has primarily focused on demonstrating the efficacy of geo-intelligence for adaptive streaming. As such, they have used simplistic averaging techniques, such Exponential Weighted Moving Average (EWMA) for creating bandwidth statistics. In this paper, we propose to use more advanced learning strategies for creating such maps. In particular, we incorporate a Neural Network (NN) model for estimating the bandwidth at each location based on historical bandwidth data. To study the efficacy of this idea, we use real mobile bandwidth traces collected in the experiments conducted in [9]. The researchers measured the downlink mobile bandwidth at approximately every 200m while driving along a particular route in the Sydney Metropolitan area. The route is 22 Km long and typical drive time ranges from 22 to 30 minutes. The bandwidth measurements were tagged with the GPS coordinates and time. 70 repeated trips were made along this route, resulting in a total of 70 bandwidth traces. Measurements were conducted in parallel for 2 HSDPA providers. In this paper, we use the traces from one provider. We used the data from the first 40 trips for training the NN and the remaining 30 trips for the evaluations.

To prepare the input data for the NN, we divide the route into 300m location segments. This gives us 73 unique locations along the route. If there are multiple bandwidth readings recorded in a location, we simply compute their average to represent the bandwidth for that location. We used a feed-forward neural network with the back-propagation algorithm to effectively learn the weights in the network during the training process. The network contains more than one hidden layer containing a number of perceptrons in each layer to improve the estimation accuracy. We consider the number of hidden layers as 4 and the number of perceptron in each hidden layer as 15, as we found that these values gave the best estimated bandwidth. We used the NN toolbox in Matlab to simulate the model [11], [12].

To evaluate the performance of the NN estimation model, we compared it with EWMA and Markov Models. We use RMSE (root mean squared error) as the evaluation metric. Given the actual bandwidth ($BW_{act_{l_i}}$) and estimated bandwidth ($BW_{est_{l_i}}$), the RMSE at any location, i_i is calculated as follows. where, N denotes the number of trips used for

$$RMSE_{l_i} = \sqrt{\frac{\sum_{j=1}^{N}(BW_{act_{l_i}} - BW_{est_{l_i}})^2}{N}}$$

evaluations. Recall that, N=30 in our evaluations. The mean of the RMSE is then estimated by averaging the RMSE of all the locations along the trip. Fig.1 shows the mean RMSE error for the 3 strategies.

As is evident from Fig.1, the NN model is clearly superior to the other two models in estimating the bandwidth. The reason is that unlike EWMA and the Markov model, given the initial training data and the appropriate learning algorithm, the NN model has the ability to naturally learn and adapt itself with the new bandwidth observations. This allows the NN model to capture a more realistic pattern of the system by continuously feeding back the errors between the estimated and actual bandwidth into the model. From the bandwidth trace files, we found the link bandwidth fluctuates not only over the locations but also over different trips. While other models cannot adapt dynamically to such changes, the NN model, due to its inherent robustness, is able to estimate the bandwidth more accurately.

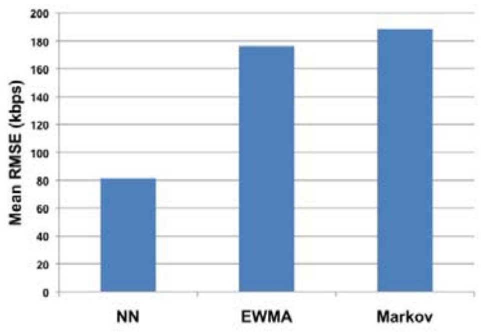

Fig. 1. Comparison of mean RMSE

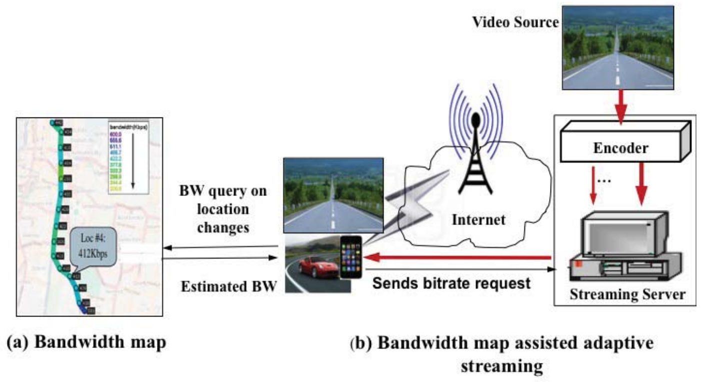

Fig. 2. Architecture of the window-based geo-intelligent HTTP Adaptive video streaming

IV. Window-based geo-intelligent HTTP Adaptive Video Streaming

In this section, we present a detailed overview of our proposed window-based geo-intelligent HTTP adaptive video streaming algorithm.

Fig.2 illustrates the system architecture, which is consistent with other geo-intelligent schemes such as [5]. We assume that the client on the mobile device stores the bandwidth map, which is generated using the neural network model discussed in Section III. We also assume that the client is aware of the route taken by the vehicle. This can be readily obtained from the in-car navigation system in the case of private cars, or from publicly available route information for public transport vehicles. Finally, we assume that the client is aware of the current location and speed of the vehicle, which can be obtained from in embedded GPS device in the mobile device or from the in-car navigation system. The client first determines the estimated bandwidth for the next LW locations from the bandwidth map. It next estimates the total time spent in each location, based upon the current speed of the vehicle and the granularity of each location segment in the map (300m in our case). By multiplying these two values, the client can now estimate the total downloadable data in the current location window LW, denotes as DD_{LW}.

Note that, in a typical streaming application, the client is initially provided with a description file that contains information about the average bitrates for the different quality streams. Based on this information, the client can estimate the total data that would need to be downloaded for each quality stream, i, denoted as $BD_{(LW,i)}$. The client can now estimate the buffer occupancy after downloading the next chunk of quality i, based on DD_{LW}, $BD_{(LW,i)}$ and the current buffer occupancy, $BuffOcpcy_{cur}$. The client then selects the quality, i, that would lead to the minimum buffer occupancy. When the client chooses the quality, i, for minimum buffer occupancy, it delivers the possible maximum quality without having any buffer underrun for this chunk. While doing so, the client disregards, the qualities that can lead to negative buffer occupancy, since these qualities are likely to cause buffer underrun. Note that, this algorithm is repeatedly executed by the client for each chunk. The pseudo-code for the algorithm is presented in Fig.3.

Intuitively, considering a small value for the location window, may not be sufficient in dealing with sharp fluctuations in bandwidth (e.g., sudden outage) which may arise at latter locations. On the other hand, if the *LW* is very large, then the impact of potential errors in bandwidth estimation is compounded, which can lead to incorrect bitrate selection decisions, leading to sub-optimal streaming performance. We will investigate this trade-off by observing the impact of varying the size of the *LW* on the streaming performance in Section V.

```
1:  Input: $G = (Loc_{Cur}, EstTime, EstBW, BuffOcpcy_{cur})$
2:  **for all** $Quality_i$ **do**
3:      **for all** $L \in Loc_{Cur} \cdots Loc_{LW}$ **do**
4:          $DD_{LW}$ = $EstTime_L \times EstBW_L$
5:          $BD_{LW}$ = $EstTime_L \times BitrateAvg_i$
6:      **end for**
        $(BuffOcpcy_{cur})_i$ = $BuffOcpcy_{cur}$ + $DD_{LW}$ - $BD_{LW}$
7:      **if** $(BuffOcpcy_{cur})_i < 0$ **then**
8:          $(BuffOcpcy_{cur})_i$= newValue($\infty$)
9:      **end if**
10: **end for**
11: Find $\min(BuffOcpcy_{cur})_i$
12: $Quality$= $i$
```

Fig. 3. Pseudo-code of the proposed window-based algorithm

where, $i = 1, 2.......n$ represents different quality streams.

LW = size of the location window

$BuffOcpcy_{cur}$ = current buffer occupancy

$EstTime_L$ = estimated location interval for location L.

$EstBW_L$ = estimated downlink bandwidth for location L

$DD_{LW,i}$ = total downloadable data that the client can download for this LW

$BD_{LW,i}$ = total data that needs to be downloaded for quality i for the LW

Although the NN model presented in Section III is more accurate at estimating bandwidth than other strategies, it can never be 100% accurate. Moreover, if the driving speed changes drastically, the estimated duration spend by the car in the location window can be inaccurate. These inaccuracies may lead to improper decisions, which can cause buffer underrun during the playback. To avoid this, we have incorporated the plain-vanilla HTTP adaptive streaming with our proposed mechanism as a fall-back policy, if the buffer occupancy becomes extremely small (8 sec in our implementation).

V. Trace-Driven Evaluations

In this section, we present the results from trace-driven evaluations. We first describe the details of the evaluation setup and then present the results for different location window parameters. We also compare the results with plain vanilla adaptive HTTP streaming.

A. Evaluation Setup

The goal of this study is to evaluate the performance of the window-based look-ahead strategies under real-world bandwidth conditions, in high-speed vehicular mobility. We consider the scenario that a passenger in a vehicle watches a streaming video on his mobile device (phone, tablet or laptop) while driving from location A to B. We assume that the viewer watches the video for the entire trip.

We have implemented an HTTP adaptive video streaming client-server prototype using JAVA. Fig.4 shows the experimental setup. Three Linux (Ubuntu 11.04) machines have been used for this experiment. The HTTP server and video files are hosted at the server machine. The bandwidth shaper emulates the bandwidth changes of the HSDPA link according to the bandwidth trace files. We use the bandwidth traces discussed in Section III for the experiments. For creating the bandwidth maps, we use the Neural Network model in Section III. Bandwidth measurements from the first 40 trips are used to precompute the bandwidth map, which is stored at the client. Since the wired links in the Internet and the HSDPA core network have sufficiently high bandwidth and small delays as compared to the last-hop HSDPA link, the server and the bandwidth shaper are connected via a static 100 Mbps Ethernet link with a 10 ms propagation delay to represent the wired Internet. The client and the bandwidth shaper are also physically connected with a 100 Mbps Ethernet link. However, in order to emulate the bandwidth changes as the vehicle travels along its route, we use the system utility *tc* at the bandwidth shaper to throttle the bandwidth of the link between the bandwidth shaper and client. In each experiment run, we emulate one driving trip, wherein the bandwidth shaper varies the bandwidth at each location according to the corresponding empirical bandwidth trace for the trip.

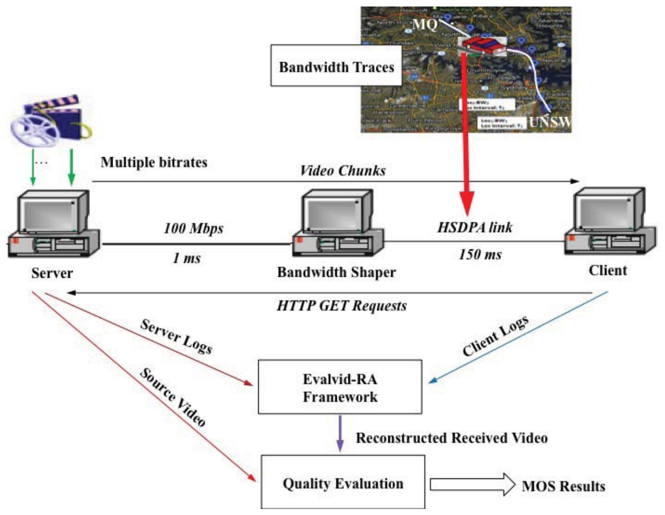

Fig. 4. Experimental setup

Further, we have implemented the *location window* based HTTP client discussed in Section IV. After a session established, the sever sends a manifest file to the client that specifies the available bitrates (typically 4-5) for each chunk [1], [13]. For our experiment, we have set the initial playout buffer as 8 sec based on experimental results in [3].

We create a looping video using a medium motion media sequence ("Foreman" QCIF sequence) encoded at 30 frames/second [14]. Before the experiment starts, the video is pre-encoded at 5 different qualities. Table I lists the average bitrate for each quality. Each encoded video streams is fragmented into a sequence of 2 second chunks.

TABLE I
VIDEO QUALITY LEVELS

Quality Level	Averag bitrate
1	240 Kbps
2	480 Kbps
3	720 Kbps
4	960 Kbps
5	1240 Mbps

To evaluate the video quality, we use the widely used Evalvid-RA framework [14]. According to the framework, we log all the required information at the server and client during the streaming session. These logs are used for reconstructing the received video sequence after finishing the experiment for a particular trip. Note that, to deal with the lost frames, we use the last played frame when playback is paused, which is a strategy followed by current streaming clients. The source and received videos are processed to generate the Peak Signal-to-Noise Ratio (PSNR) for each frame [14].

B. Results

In this section, we present the results from the trace-driven experiments. Recall that, the goal of the experiments is to investigate the impact of varying the location window on the user-perceived performance using the aforementioned metrics. In particular, we are interested to know if there is an optimal value for the location window that provides the best viewing experience. For this, we undertake a parametric study by varying the location window size from 1 at one extreme to the entire trip (73 locations for the traces used) at the other extreme. Rather than presenting results for each and every value of the window, we only include data for window sizes of 5, 10 and 20 in addition to the two extremes, since these are sufficient to reveal the general trend. When the number of remaining locations along the route becomes less than the size of the window, our algorithm uses the remaining locations as the window size. Recall that, when the location window is 1, our algorithm is reduced to the greedy strategy proposed in [5]. Similarly, at the other extreme, when we consider the entire remainder of the route, the strategy resembles the one proposed in [6]. We also include plain-vanilla adaptive HTTP streaming in our comparisons, to serve as a benchmark. For this, we implemented the basic reactive strategy employed by adaptive HTTP within the evaluation framework discussed earlier.

Figs. 5 and 6 show the effect of using different values of the *LW* on the number of buffer underruns (i.e. freezing occurrences) and the cumulative duration of buffer underruns, respectively. These graphs illustrate an interesting trend, in that, the performance is worse at the two end-points. Moreover, the best performance is observed when the window size is 5. The performance worsens as the window size is increased beyond this optimal value of 5. This trend can be explained as follows. First, when the window size is 1, the client only factors in the estimated bandwidth at the next location in its decision making process. As such, if there are significant fluctuations (e.g., outages) at some of the upcoming locations, this strategy cannot plan well in advance for such events and pre-buffer sufficient data to cope with the expected fluctuations. Second, at the other extreme, the strategy will take into account the estimated bandwidth and the estimated dwell time in all remaining locations for selecting the streaming quality.

Given that these estimations are never 100% accurate, such a large look-ahead window means that the likelihood of estimation errors is not only higher but also that these errors are compounded. As a result, on occasion, improper choices are made in selecting the quality of the next chunk, which translates to increased instances of buffer underruns. A window size of 5 seems to strike a good balance between the two. In that instance, it is able to absorb any significant upcoming bandwidth fluctuations by ensuring there is always data in the buffer, while also minimising the impact of estimation errors. Observe that, on average, the number of instances when the buffer is empty is less than 0.5. This means that in more than half of the trips used in the evaluations, no buffer underrun was experienced. Even in the worst case, only one such event was experienced in a trip.

Furthermore, frequent quality changes in adaptive streaming is annoying for users, particularly when the length of transition is less than 2 seconds [15], [16]. As such, we include this metric in our evaluations. We quantify the frequency of quality changes on a chunk-by-chunk basis.

Table II compares the mean results of various parameters for

TABLE II
EFFECT OF VARYING THE LOCATION WINDOW

location window (LW)	LW= 1	LW= 5	LW= 10	LW= 20	LW= 73	Plain HTTP
Buffer Size(Sec)	8	8	8	8	8	8
Number of buffer underruns	2.33	0.33	0.83	1.17	2.67	3.80
Cumulative duration of buffer underruns (Sec)	49.82	4.98	13.65	21.94	48.91	66.23
Frequency of quality change(%)	28.44	16.24	13.46	13.27	22.22	42.91
Average PSNR(dB)	35.12	34.98	34.87	34.89	35.09	34.92

different values of the location window for the window-based geo-intelligent approach and plain-vanilla HTTP streaming. The trend observed in the earlier results is consistent, where the window size of 5 achieves the best results. In comparison with plain HTTP streaming, this optimal window size reduces the number of buffer underruns, the cumulative duration of underruns and the quality fluctuations by 92.11%, 92.48% and 62.15%, respectively.

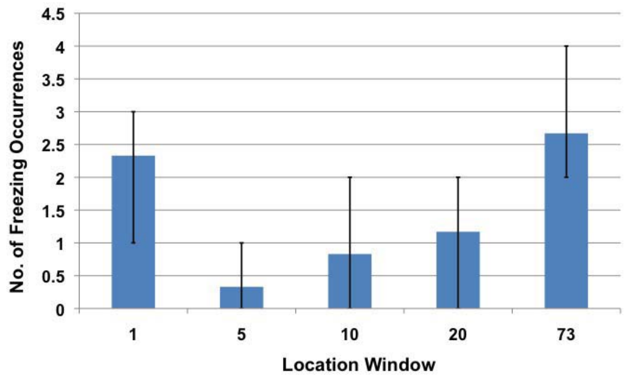

Fig. 5. Effect of location window on the number of freezing occurrences

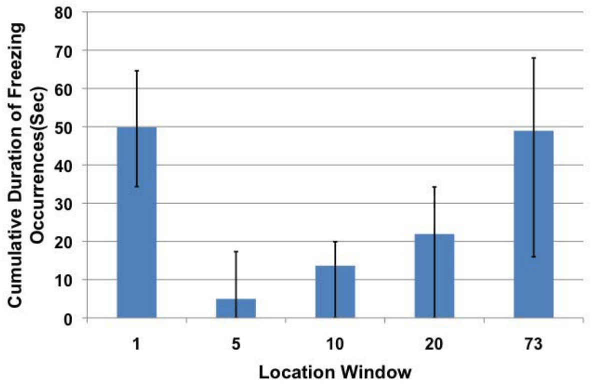

Fig. 6. Effect of location window on the cumulative duration of freezing occurrences

The above discussion confirms that the geo-intelligent mechanism with an optimum *location window* can achieve near an uninterrupted video streaming session and thus improves the QoE in high-speed mobility.

VI. CONCLUSION

In this paper, we have undertaken a systematic evaluation of look-ahead strategies for geo-intelligent adaptive HTTP streaming in high-speed mobility. We proposed a *location window* based geo-intelligent algorithm that utilizes the estimated bandwidth along the next X locations. The emphasis is on the systematic study to investigate the effect of varying the location window on the streaming experience of the mobile users. Our simulation results using real-world mobile bandwidth traces show that an optimum location window size of 5 provides the best streaming performance. For a majority of trips, the optimal strategy entirely eliminates buffer underruns. We also present a neural network model for creating bandwidth maps and demonstrate its efficacy in estimating bandwidth.

REFERENCES

[1] A. Zambelli, "Iis smooth streaming technical overview," *Microsoft Corporation*, 2009.
[2] F. P. Tso, J. Teng, W. Jia, and D. Xuan, "Mobility: A double-edged sword for hspa networks," in *Proceedings of ACM MobiHoc 2010*. ACM, 2010.
[3] J. Yao, S. Kanhere, I. Hossain, and M. Hassan, "Empirical evaluation of http adaptive streaming under vehicular mobility," *NETWORKING 2011*, pp. 92–105, 2011.
[4] X. Qiu, H. Liu, D. Li, S. Zhang, D. Ghosal, and B. Mukherjee, "Optimizing http-based adaptive video streaming for wireless access networks," in *Broadband Network and Multimedia Technology (IC-BNMT), 2010 3rd IEEE International Conference on*. IEEE, 2010, pp. 838–845.
[5] J. Yao, S. Kanhere, and M. Hassan, "Improving qos in high-speed mobility using bandwidth maps," *Mobile Computing, IEEE Transactions on*, no. 99, pp. 1–1, 2011.
[6] H. Riiser, T. Endestad, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Video streaming using a location-based bandwidth-lookup service for bitrate planning," *ACM TOMCCAP*, 2011.
[7] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate adaptation for adaptive http streaming," *Proc. ACM MMSys*, 2011.
[8] D. Jarnikov and T. Özçelebi, "Client intelligence for adaptive streaming solutions," *Signal Processing: Image Communication*, 2011.
[9] J. Yao, S. Kanhere, and M. Hassan, "An empirical study of bandwidth predictability in mobile computing," in *Proceedings of the third ACM international workshop on Wireless network testbeds, experimental evaluation and characterization*. ACM, 2008, pp. 11–18.
[10] I. Curcio, V. Vadakital, and M. Hannuksela, "Geo-predictive real-time media delivery in mobile environment," in *Proceedings of the 3rd workshop on Mobile video delivery*. ACM, 2010, pp. 3–8.
[11] H. Demuth, M. Beale, and M. Hagan, "Neural network toolboxł 6," *User Guide© COPYRIGHT*, vol. 2009, 1992.
[12] S. Haykin and N. Network, "A comprehensive foundation," *Neural Networks*, vol. 2, 2004.
[13] R. Pantos, "Http live streaming," 2011.
[14] A. Lie and J. Klaue, "Evalvid-ra: trace driven simulation of rate adaptive mpeg-4 vbr video," *Multimedia Systems*, vol. 14, no. 1, pp. 33–50, 2008.
[15] P. Ni, A. Eichhorn, C. Griwodz, and P. Halvorsen, "Fine-grained scalable streaming from coarse-grained videos," in *Proceedings of the 18th international workshop on Network and operating systems support for digital audio and video*. ACM, 2009, pp. 103–108.
[16] M. Zink, O. Künzel, J. Schmitt, and R. Steinmetz, "Subjective impression of variations in layer encoded videos," in *Proceedings of the 11th international conference on Quality of service*. Springer-Verlag, 2003, pp. 137–154.