Multi-Source Mobile Video Streaming: Load Balancing, Fault Tolerance, and Offloading with Prefetching*

Dimitris Dimopoulos, Christos Boursinos, and Vasilios A. Siris

Mobile Multimedia Laboratory, Department of Informatics
Athens University of Economics and Business, Greece
vsiris@aueb.gr

Abstract. We present the design and experiments from a testbed implementation of multi-source mobile video streaming that combines three mechanisms: 1) load balancing among different paths from multiple sources, 2) resilience to link and server failures, and 3) enhanced offloading by exploiting mobility and throughput prediction to prefetch video data in caches located at hotspots that the mobile will encounter. Our testbed consists of an Android mobile video streaming client that can utilize both cellular and Wi-Fi interfaces and request different parts of a video from different servers, a server that accepts client requests for parts of a video, and a cache server that accepts client requests to proactively fetch parts of a video so that they are immediately available when the mobile client enters the cache server's hotspot.

1 Introduction

A major trend in mobile networks over the last few years is the exponential increase of powerful mobile devices, such as smartphones and tablets, with multiple heterogeneous wireless interfaces that include 3G/4G/LTE and Wi-Fi. The proliferation of such devices has resulted in a skyrocketing growth of mobile traffic, which in 2013 grew 81%, becoming nearly 18-times the global Internet traffic in 2000, and is expected to grow 10-fold from 2013 until 2018[2]. Moreover, mobile video traffic was 53% of the total traffic by the end of 2013 and is expected to be over two-thirds of the world's mobile data traffic by 2018. The increase of video traffic will further intensify the strain on cellular networks, hence reliable and efficient support for video traffic in future networks will be paramount.

Efficient support for video streaming in future mobile environments, in terms of both network resource utilization and energy consumption, will require integration of heterogeneous wireless technologies with complementary characteristics; this includes cellular networks with wide-area coverage and Wi-Fi hotspots with high throughput and energy efficient data transfer. Indeed, the industry has already verified the significance of mobile data offloading to exploit fixed

* This research has been co-financed by the European Union (European Social Fund-ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF)-Research Funding Program: Aristeia II/I-CAN.

[2] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2013-2018, Feb. 5, 2014

broadband and Wi-Fi technology: globally, 33% of total mobile data traffic was offloaded onto Wi-Fi networks or femtocells in 2012[1].

The contribution of this paper is to present the design and experiments from the testbed implementation of a system for multi-source mobile video streaming that combines functions for load balancing and fault tolerance, in addition to implementing an innovative procedure for enhanced mobile data offloading that utilizes mobility and throughput prediction to prefetch video data in local caches at hotspots that a mobile will encounter. Indeed, prior work has verified that mobility and throughput prediction is possible; this paper is not concerned with developing a system for such prediction, but rather focuses on an actual implementation of mechanisms that exploit such prediction. The work in this paper is different from our previous work in [13, 11] that considers mobile data offloading for delay tolerant traffic, which requires transferring a file within a time threshold, and delay sensitive traffic, which requires minimizing the file transfer time; unlike these traffic types, video streaming requires a continuous transfer of video data to avoid impact on a user's QoE (Quality of Experience), thus necessitates a totally different prefetching procedure and evaluation. Also, unlike [12] which contains trace-driven simulation, here we focus on the design and experiments from an actual implementation of enhanced offloading using prefetching, which is combined with mechanisms for load balancing and fault tolerance.

The rest of the paper is structured as follows: In Section 2 we present related work. In Section 3 we present the design of the multi-source mobile video streaming system and in Section 4 we present the mechanisms for load balancing, fault tolerance, and enhanced offloading using prefetching. In Section 5 we present experiments that illustrate the behavior of the system and the gains in terms of increased mobile data offloading and improved QoE.

2 Related work

Prior work has demonstrated bandwidth predictability for both cellular networks [16] and Wi-Fi [7]. Bandwidth prediction for improving video streaming is investigated in [17, 4], and for client-side pre-buffering to improve video streaming in [10]. The work in [17, 4, 10] focuses on cellular networks, whereas we consider integrated cellular and Wi-Fi networks. Moreover, our goal is not to develop a new system for mobility and bandwidth prediction, but to exploit such prediction to prefetch data in order to improve mobile video streaming.

Multi-source video streaming for improving robustness in mobile ad hoc networks is investigated in [9], which focuses on video and channel coding. The work in [2] investigates joint routing and rate allocation for multi-source video streaming in wireless mesh networks. Load balancing over multiple radio interfaces is investigated in [3], which focuses on client-side scheduling. [14] investigates load balancing by probabilistically splitting a video flow across multiple radio interfaces based on video transmission patterns. The adaptation of P2P techniques for multi-source video streaming to Android clients is investigated in [8].

The feasibility of using prediction for prefetching is investigated in [1], which however does not propose or evaluate specific prefetching algorithms. Prefetching

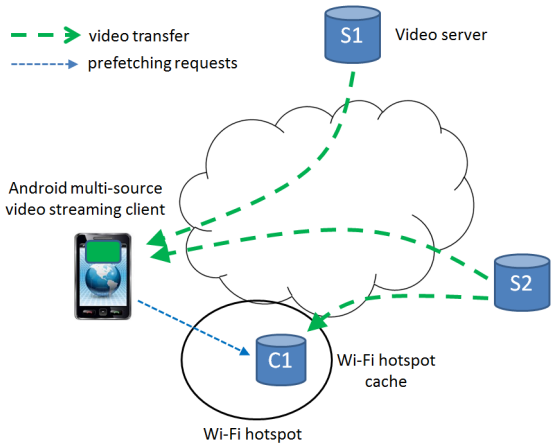


Fig. 1. The system architecture consists of an Android multi-source mobile video streaming client, video servers, and local hotspot caches for prefetching video data.

for improving video file delivery in cellular femtocell networks is investigated in [5], and to reduce the peak load of mobile networks by offloading traffic to Wi-Fi hotspots in [6]. Our work differs from the above work on multi-source streaming and prefetching in that it presents an actual testbed implementation of multi-source mobile video streaming that combines mechanisms for load balancing, fault tolerance, and an innovative procedure for prefetching video data in Wi-Fi hotspots that the mobile will encounter in order to improve video streaming.

3 System design

The system consists of i) an Android client running in a mobile device that can playback a video while streaming different parts of the video from multiple servers, ii) video servers that accept requests for parts (chunks) of a video, and iii) caches located in Wi-Fi hotspots that accept requests from mobile clients to prefetch chunks of a video from a remote server, Figure 1. Next we describe in more detail each of these three entities.

The multi-source mobile video streaming client contains all the intelligence for downloading parts of a video file from multiple servers. In particular, the video streaming client implements the following three procedures:

- load balancing: the client measures the throughput that it receives data from different video servers, and adjusts the number of video chunks that it requests from each server based on the measured throughput.
- fault tolerance: the client can detect when a server or the path from a server is down, and request video chunks from another available server.
- enhanced offloading with prefetching: the client exploits mobility and throughput prediction to send to local caches in hotspots that it will encounter requests to prefetch parts of the video, so that they are immediately available when the mobile device connects to these hotspots.

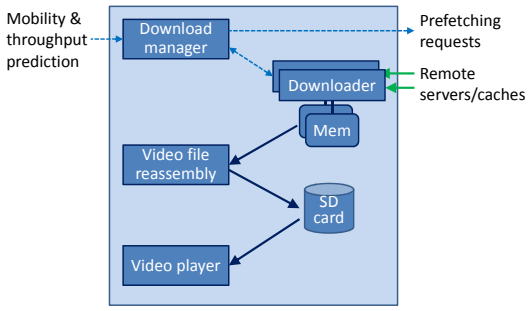


Fig. 2. Android multi-source mobile video streaming client design.

The high-level design of the multi-source video client is shown in Figure 2. The download manager obtains mobility and throughput prediction information, based on which it instructs a local cache in the Wi-Fi hotspot that the mobile will encounter to prefetch video data. The video is segmented into multiple chunks that are contained in separate files. Each chunk is transferred to the mobile client through a separate TCP connection; this is performed by the downloader modules, Figure 2, where each downloader is responsible for transferring video data from a particular server. Such an approach for TCP-based video streaming, by breaking the video into multiple chunks, is used in the MPEG-DASH standard. However, we did not use the MPEG-DASH video standard because at the time of our implementation there was no stable MPEG-DASH video player for Android. Nevertheless, the design of the multi-source mobile video streaming client and the procedures implementing the aforementioned functionality are independent of the details of the protocol used for transferring video chunks.

To download video from multiple servers, the mobile client needs to know the IP addresses of these video servers, which can be included in the mobility prediction information or in metadata files such as MPEG-DASH's Media Presentation Description (MPD). Alternatively, knowledge of the video servers' IP addresses is not necessary in Information-Centric Network (ICN) architectures, where users request content based on the name for the content [15].

Unlike the multi-source mobile video streaming client which implements load balancing, fault tolerance, and prefetching, the video server simply accepts requests for video chunks, which are stored locally in separate files. Cache servers located in Wi-Fi hotspots are involved in video transfer only when prefetching is used. The advantages of prefetching are that transferring video data from a local hotspot cache can fully utilize the Wi-Fi throughput, which is typically higher than the backhaul throughput that connects the hotspot to the Internet. If the video is not prefetched, then the amount of video data transferred through the Wi-Fi hotspot is constrained by the available backhaul throughput.

4 Mechanisms

In this section we describe in more detail the three mechanisms implemented in the multi-source mobile video streaming client.

4.1 Load balancing

This mechanism balances the load among the available servers, based on the throughput from each server. Specifically, the transfer of a video file occurs in rounds. In each round a specific number of video chunks C are transferred, which depends on the video playout rate R_{playout} and the chunk size S. In particular, the number of chunks C should satisfy $C \cdot S \geq R_{\text{playout}} \cdot T$, where T is the time for transferring C chunks, and depends on the number of servers and the throughput from each server. Let r_i be the throughput from video server i measured in one round. If N is the number of servers, then the number of chunks c_i transferred from server i in the next round is given by

$$c_i = \frac{r_i}{\sum_{j=1}^{N} r_j} C \text{ for } i = 1, ..., N-1 \quad \text{and} \quad c_N = C - \sum_{i=1}^{N-1} c_i \,.$$

The throughput from each video server is measured by the downloaders, whereas the calculation of the number of video chunks that are requested from each server is performed in the download manager, Figure 2.

4.2 Fault tolerance

The fault tolerance mechanism detects when a video server or a path from a server is down, in which case it downloads video chunks from an alternative server. Detection of a server or path fault is performed by the downloaders based on both a timeout (set to 50 milliseconds) for creating a new TCP connection and a broken TCP connection. Moreover, to handle the case of transient failures, the client periodically requests video data from a server that was previously down, allowing it to detect when the server becomes operational again.

4.3 Enhanced offloading with prefetching

Mobility prediction provides knowledge of how many Wi-Fi hotspots a mobile will encounter, when they will be encountered, and for how long the node will be in each hotspot's range. In addition to this mobility information, we assume that information on the estimated throughput in the Wi-Fi hotspots and the cellular network is also available; for the former, the information includes both the throughput for transferring data from a remote location, e.g., through an ADSL backhaul, and the throughput for transferring data from a local cache.

The procedure to exploit mobility and throughput prediction for prefetching is shown in Algorithm 1, which is implemented in the download manager of the multi-source mobile video streaming client. The algorithm extends the one investigated using trace-driven simulation in [12], by exploiting knowledge of the video buffer playout rate to reduce the throughput which it downloads video data over the mobile network. The procedure defines the mobile's actions when it exits a Wi-Fi hotspot, hence has only mobile access (Line 9), and when it enters a Wi-Fi hotspot (Line 14). Mobility and throughput prediction allows the mobile to determine when it will encounter the next Wi-Fi hotspot that has higher throughput than the cellular network's throughput. From the time to reach the next hotspot and the average video buffer playout rate, the mobile can

estimate the position that the video stream is expected to reach (CurrentPosition + Offset) when it arrives at the next Wi-Fi hotspot (Line 10). It then sends a request to the cache in the next hotspot it will encounter to start caching video data from that position (Line 11). The video buffer playout rate is also used to estimate the throughput at which it should download video data while in the mobile network (Line 12).

Algorithm 1 Using mobility and throughput prediction to prefetch video data

1: **Variables:**
2: R_{playout}: average video buffer playout rate
3: $T_{\text{next Wi-Fi}}$: average time until node enters range of next Wi-Fi
4: *CurrentPosition*: current position of video stream
5: *Offset*: estimated offset of video stream when node enters next Wi-Fi hotspot
6: B: amount of video data in buffer
7: *RateMobile*: rate at which video is downloaded from mobile network
8: **Algorithm:**
9: **if** node exits Wi-Fi hotspot **then**
10: $Offset \leftarrow R_{\text{playout}} \cdot T_{\text{next Wi-Fi}}$
11: Start caching video stream in next Wi-Fi starting from $CurrentPosition + Offset$
12: $RateMobile \leftarrow R_{\text{playout}} - \frac{B}{T_{\text{next WiFi}}}$
13: Download video data from mobile network with rate *RateMobile*
14: **else if** node enters Wi-Fi hotspot **then**
15: Transfer video data that has not been received up to *Offset* from original location
16: Transfer video data from local cache
17: Use remaining time in Wi-Fi hotspot to transfer video data from original location
18: **end if**

When the node enters a Wi-Fi hotspot, it might be missing some portion of the video stream up to the offset from which data was cached in the hotspot; this can occur if, due to time variations, the node reaches the Wi-Fi hotspot earlier than the time it had initially estimated. In this case, the missing data needs to be transferred from the video's original remote location (Line 15), through the hotspot's backhaul link. Also, the amount of data cached in the Wi-Fi hotspot can be smaller than the amount the node could download within the time it is in the hotspot's range. In this case, the node uses its remaining time in the Wi-Fi hotspot to transfer data, as above, from the video's original location (Line 17).

5 Experiments

In this section we present experimental results that illustrate the load balancing and resilience mechanisms and the performance gains of enhanced offloading with prefetching, in terms of a higher percentage of offloaded traffic and improved Quality of Experience (QoE), expressed through the reduced number of video pauses (or stalls). Our goal also includes demonstrating the flexibility provided by combining an actual multi-source mobile video streaming implementation with mobility emulation, in terms of time-varying connectivity type (mobile or Wi-Fi) and throughput, to execute experiments with different throughput values and different hotspot configurations.

5.1 Experiment setup

In the beginning of each experiment the mobile client obtains a description of the experiment scenario in an XML (Extended Markup Language) file. The file

specifies the mobile's connectivity, e.g. Wi-Fi or cellular, for different segments and the IP addresses of the video servers and caches (in the case of prefetching); the connectivity segments are specified by their starting time. Essentially, the scenario description file allows us to emulate the device's mobility, in terms of different connectivity scenarios, in addition to different maximum download rates for the mobile, Wi-Fi, and ADSL backhaul link; the latter is performed using the wondershaper network traffic shaping tool. The above mobility (in terms of time-varying connectivity type) and throughput emulation provides the necessary flexibility to perform experiments with a range of parameters and assess the performance of the system in scenarios with a different number, location, and throughput of Wi-Fi hotspots. Finally, our testbed implementation can support scenarios where both the mobile and WiFi interface are used simultaneously; this is achieved with the tethering feature of Android devices, which allows both the mobile and Wi-Fi interface to be simultaneously active.

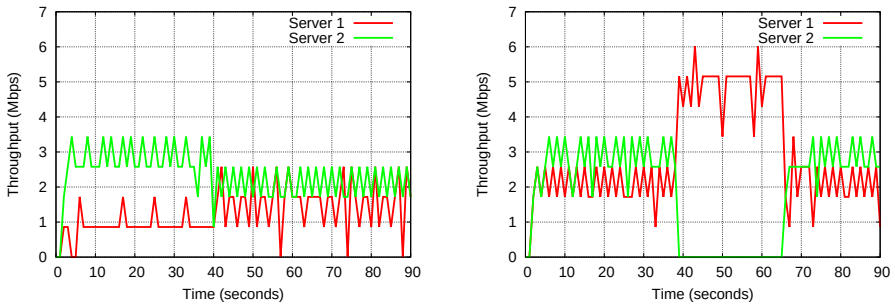
The video used in the experiments was a 596 second clip from Big Buck Bunny, encoded at 1280x720 and with an average rate of approximately 1.65 Mbps. The video was segmented into 1229 chunks, each with size approximately 97 KBytes. In the experiments the multi-source mobile video streaming client was running on a Galaxy S2 smartphone with Android 4.0.4. The video and cache servers were running on two virtual machines with Ubuntu 13.10, executed in a workstation with VirtualBox 4.3.6.

5.2 Results

Load balancing: Figure 3(a) shows the download throughput when video is streamed from two servers. Initially, the maximum throughput from each server is 1 and 3 Mbps. At approximately 40 seconds the maximum downlink rate from each server becomes 2 Mbps, and the achieved download throughput from both servers approaches this value. Of course, throughout the experiment the video is played back without any pauses (stalls).

Fault tolerance: Figure 3(b) shows the download throughput from two servers. Initially, the load is balanced among the two servers. At approximately 40 seconds the second server falls and the download throughput from the first server increases. Later, at time 65 seconds the second server becomes available again and the load is again equally distributed between the two servers.

Enhanced offloading with prefetching: The next set of experiments show the performance gains that can be achieved with prefetching, in terms of the increased percentage of offloaded traffic and the improved video QoE through the reduction of the number of frame pauses (stalls). By default each experiment involves a total of 6 Wi-Fi hotspots, which the mobile encounters at time 0, 100, 200, 300, 400, and 500 seconds. The mobile is able to download video data in each hotspot for a duration of 20 seconds; note that prefetching cannot be performed in the first hotspot, since the experiments begin when the mobile is already in the first hotspot. We also assume that there is some randomness in the time a hotspot is encountered and in the maximum download throughput in each segment. The default variability of the time and throughput is 2% and



(a) Load balancing (b) Fault tolerance

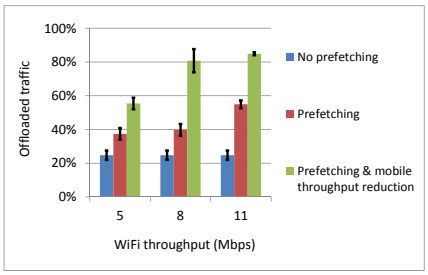
Fig. 3. Load balancing and fault tolerance.

5%, respectively, while we also present results for different variabilities. A 5% variability for throughput 1 Mbps means that the actual throughput is randomly selected from the interval $[950, 1050]$ Mbps, hence there is a mismatch between predicted and actual throughput. The graphs in this section show the average of five runs and the corresponding 95% confidence interval.

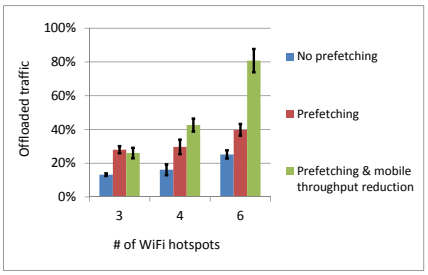
Percentage of offloaded traffic: Figure 4(a) shows the percentage of video traffic offloaded to Wi-Fi for three schemes: 1) no prefetching (i.e., when the mobile enters a hotspot the video is downloaded from a remote server using the maximum ADSL bauckhaul throughput), 2) prefetching and downloading of video data over the mobile network at the maximum rate, and 3) prefetching and downloading video data over the mobile network at a smaller rate, Algorithm 1. Observe that the percentage of offloaded traffic with prefetching increases when the Wi-Fi throughput increases, verifying that prefetching can utilize the higher Wi-Fi throughput; on the other hand, without prefetching the percentage of offloading is independent of the Wi-Fi throughput, since the ADSL backhaul is the bottleneck. Also, a higher percentage of offloading is achieved with prefetching and reduction of the mobile throughput. Note that maximum offloading percentage is approximately 85%, since the video data transferred in the second mobile segment, which is approximately 15%, cannot be offloaded.

Figure 4(b) shows the percentage of offloading for a different number of hotspots: 4 hotspots located at times 0, 100, 300, and 500 seconds, and 3 hotspots located at times 0, 100, and 300 seconds. More hotspots allow a higher percentage of offloading. Also, note that the two prefetching schemes achieve the same offloading for 3 hotspots; this occurs because when the number of hotspots is small, prefetching fully utilizes the available Wi-Fi throughput, hence Algorithm 1 uses the maximum mobile throughput. In general, the offloading gains depend on the location and duration of connectivity in hotspots. Moreover, higher values of the time variation (up to 10%) and throughput variation (up to 20%) yielded similar offloading results and are not included due to space constraints.

Improved video QoE: Next we investigate the improved QoE that can be achieved with prefetching, in terms of fewer video frame pauses. Figure 5 shows that the



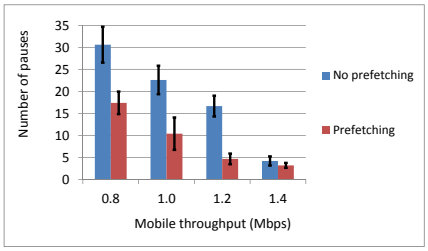
(a) Wi-Fi throughput, 6 hotspots



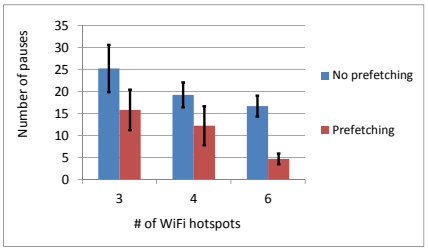
(b) # of hotspots, $R_{\text{Wi-Fi}} = 8$ Mbps

Fig. 4. Mobile video data offloading. $R_{\text{mobile}} = 2$ Mbps, $R_{\text{adsl}} = 3$ Mbps.

gains in terms of fewer pauses is higher when the mobile throughput is smaller; this is expected since more frame pauses occur when the mobile throughput is smaller, which is when the higher throughput of Wi-Fi can be utilized with prefetching to download more video data and avoid frame pauses; on the other hand, when prefetching is not used, while in a hotspot the video downloading rate is constrained by the ADSL throughput. Note that in the scenarios of this subsection traffic is downloaded over the mobile network using the maximum throughput, hence we do not differentiate between the two prefetching schemes considered in the previous subsection.



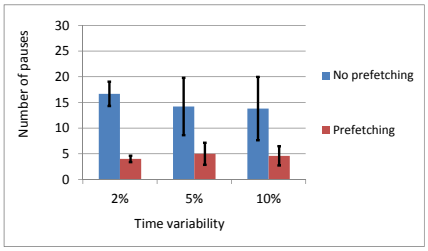
(a) Mobile throughput, 6 hotspots



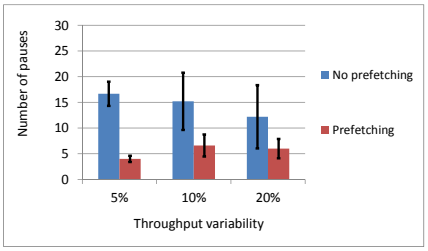
(b) # of hotspots, $R_{\text{mobile}} = 1.2$ Mbps

Fig. 5. Mobile data QoE. $R_{\text{Wi-Fi}} = 8$ Mbps, $R_{\text{adsl}} = 3$ Mbps.

Influence of time and throughput variability: Figures 6(a) and 6(b) show the QoE for different time and throughput variabilities, respectively; these figures show that the variance of the measured pauses increases with higher variability, but prefetching still achieves fewer frame pauses.



(a) Time



(b) Throughput

Fig. 6. Influence of time and throughput variability. $R_{\text{Wi-Fi}} = 8$ Mbps, $R_{\text{adsl}} = 3$ Mbps, $R_{\text{mobile}} = 1.2$ Mbps.

6 Conclusions and Future Work

We have presented a testbed implementation of multi-source mobile video streaming for integrated cellular and Wi-Fi networks that combines mechanisms for load balancing, fault tolerance, and enhanced offloading with prefetching video data in local hotspot caches. Experimental results illustrate the functionality and performance of the above mechanisms in addition to the ability of the testbed framework to execute scenarios with different connectivity types, throughput values, and hotspot configurations. Future work includes extending the implementation to investigate QoE-aware adaptation of Scalable Video Coding (SVC) streaming.

References

1. P. Deshpande, A. Kashyap, C. Sung, and S. Das. Predictive Methods for Improved Vehicular WiFi Access. In *Proc. of ACM MobiSys*, 2009.
2. Y. Ding, Y. Yang, and L. Xiao. Multi-Path Routing and Rate Allocation for Multi-Source Video On-Demand Streaming in Wireless Mesh Networks. In *Proc. of IEEE INFOCOM*, 2011.
3. K. Evensen, D. Kaspar, C. Griwodz, P. Halvorsen, A. F. Hansen, and P. Engelstad. Improving the Performance of Quality-Adaptive Video Streaming over Multiple Heterogeneous Access Networks. In *Proc. of ACM Multimedia Systems*, 2011.
4. K. Evensen, A. Petlund, H. Riiser, P. Vigmostad, D. Kaspar, C. Griwodz, and P. Halvorsen. Mobile Video Streaming Using Location-Based Network Prediction and Transparent Handover. In *Proc. of ACM NOSDAV*, 2011.
5. N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire. FemtoCaching: Wireless Video Content Delivery through Distributed Caching Helpers. In *Proc. of IEEE Infocom*, 2012.
6. F. Malandrino, M. Kurant, A. Markopoulou, C. Westphal, and U. C. Kozat. Proactive Seeding for Information Cascades in Cellular Networks. In *Proc. of IEEE Infocom*, 2012.
7. A. J. Nicholson and B. D. Noble. BreadCrumbs: Forecasting Mobile Connectivity. In *Proc. of ACM Mobicom*, 2008.
8. U. R. K. P. M. Eittenberger, M. Herbst. RapidStream: P2P Streaming on Android. In *Proc. of 19th IEEE Intl Packet Video Workshop*, 2012.
9. T. Schierl, K. Ganger, C. Hellge, T. Wiedand, and T. Stockhammer. Svc-based miltisource streaming for robust video transmission in mobile ad hoc networks. *IEEE Wireless Communications*, pages 96–103, October 2006.
10. V. Singh, J. Ott, and I. Curcio. Predictive Buffering for Streaming Video in 3G Networks. In *Proc. of IEEE WoWMoM*, 2012.
11. V. A. Siris and M. Anagnostopoulou. Performance and Energy Efficiency of Mobile Data Offloading with Mobility Prediction and Prefetching. In *Proc. of IEEE Workshop on Convergence among Heterogeneous Wireless Systems in Future Internet (CONWIRE), co-located with IEEE WoWMoM*, 2013.
12. V. A. Siris, M. Anagnostopoulou, and D. Dimopoulos. Improving Mobile Video Streaming with Mobility Prediction and Prefetching in Integrated Cellular-WiFi Networks. In *10th Int'l Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (Mobiquitous)*, 2013.
13. V. A. Siris and D. Kalyvas. Enhancing Mobile Data Offloading with Mobility Prediction and Prefetching. In *Proc. of ACM MOBICOM MobiArch Workshop*, 2012.
14. W. Song and W. Zhuang. Performance analysis of probabilistic multipath transmission of video streaming traffic over multi-radio wireless devices. *IEEE Transactions on Wireless Communications*, 11(4):1554–1564, April 2012.
15. G. Xylomenos, X. Vasilakos, C. Tsilopoulos, V. A. Siris, and G. C. Polyzos. Caching and Mobility Support in a Publish-Subscribe Internet Architecture. *IEEE Comm. Mag.*, 50(7):128–136, 2012.
16. J. Yao, S. S. Kahnere, and M. Hassan. An Empirical Study of Bandwidth Predictability in Mobile Computing. In *Proc. of ACM WinTech*, 2008.
17. J. Yao, S. S. Kahnere, and M. Hassan. Quality Improvement of Mobile Video Using Geo-intelligent Rate Adaptation. In *Proc. of IEEE WCNC*, 2010.