

Efficient Real Time Content Delivery on Wireless Networks

Ramana Isukapalli

Alcatel-Lucent

67, Whippany Road, Whippany, NJ 07981, USA

risukapalli@alcatel-lucent.com

Abstract—Wireless networks are very popular today; however, the bandwidth over the air interface is limited to support many real time applications like video on demand. While some of these applications require high quality picture frames with high resolution, large image size, etc., many others (like remote surveillance) need simple and continuous, yet intelligible image frames that are not jittery. A video server that has an estimate of the available bandwidth can stream high quality data if the bandwidth is available, or stream data that needs lower bandwidth (perhaps with a lower size, lower frame rate, etc.) if the available bandwidth is limited. Unfortunately, that poses two issues — first, the video server should be aware of the available bandwidth. Second, it should be able to modify the content it streams, “dynamically”.

In this paper, we do a detailed analysis of some popular “controlled lossy compression” methods that are used to reduce the overall “entropy” of the source data (video stream), leading to a smaller data size and reducing the overall bandwidth requirements. We discuss these methods in the context of image processing techniques and show how they can modify the data of stored video content, as needed, to present an intelligible video stream that is smaller in size. We present an end to end architecture that (i) uses “RTCP” to get an estimate of the available channel bandwidth and (ii) uses a video server that can dynamically change the parameters like image size and frame rate to produce video streams that are best suited for the available bandwidth. We present theoretical compression rates for each of these compression methods and present empirical results to confirm our claims.

I. INTRODUCTION

Data transmission over wireless networks has been gaining importance rapidly for the past few years. A number of applications ranging from accessing stock quotes and email to web surfing, etc. using mobile devices like cell phones, PDAs and laptops require some form of data transmission. Unfortunately, unlike wireline networks, air interface becomes a major bottleneck to transmit data in wireless networks. The deployment of 3G networks like UMTS and EVDO can ease this problem somewhat, by providing higher transmission rates ranging from about 144 Kbps to 7 Mbps. But these networks are still not quite widespread as of today, and very rarely can they provide the maximum data rates supported by the standards. Moreover, the available bandwidth is shared among multiple users, thereby reducing the actual bandwidth available to any particular user.

Quality of service (QoS) is an important factor that has to be taken into account for real time applications like streaming video. QoS is being given special consideration in many wireless network standards like EVDO and WiMAX, and there are many approaches [3], [7] that address this issue. However, assuring end to end quality is not always guaranteed in many situations. The best quality an end user receives is as good as the weakest link in the path the data packets take from source (say, a video server) to destination (end user). Often, the path is dynamic that depends on several factors like congestion, availability of routers on the path, etc. The limited bandwidth on air

interfaces makes the problem worse on wireless networks for data intensive real time applications.

Video streaming over data networks has been on the rise recently, it is in fact a reality over wireline data networks today. It is data intensive and can be difficult to support on wireless networks. While many of the video streaming applications require high quality picture frames, with high resolution and clarity in the video data (as in the case of movie clips), many others do not. Consider a case where a user is watching her child with a babysitter remotely, or a remote surveillance security agency monitoring properties of their clients in various locations. Clearly, the quality of the picture is of secondary importance in these situations; it is sufficient if the picture is *intelligible* and *continuous*. That is, transmitting a video stream continuously, using picture frames that are comprehensible, though of low quality (e.g., in greyscale or in relatively smaller size) serves the purpose better than a high quality stream that requires higher bandwidth and can potentially result in discontinuous stream and jitter (noise) to the receiver. Of course, if the bandwidth is indeed available, a high quality picture is definitely preferred, be it for watching movie clips or remote surveillance or any other application. Or, some users may always insist on having only the best picture without any degradation in size or quality. Unfortunately, this poses some challenges — first, the video server needs to be aware of the bandwidth availability and second, it should be able to change the format of the content *dynamically* as required, *i.e.*, to a smaller (or larger) size, grey scale (or color) and so on.

In this work we address the issues given above. We use “Realtime Protocol” (RTP) [5], an application layer protocol that usually runs over UDP, to transmit video streams. We present an end to end architecture where a video server is given feedback of the channel capacity and bandwidth from the clients by using “RTP Control Protocol” (RTCP). We use RTCP feedback and address the QoS issue — *i.e.*, the video server uses this feedback information and *dynamically* changes the source content before it passes through a video encoder, to account for varying bandwidth. Our approach is *not* an alternative to many other approaches that address the QoS issues. In fact, we can use our approach *in addition* to many other approaches (e.g., resource allocation) that address QoS issues.

The rest of this paper is organized as follows. Section II discusses data compression and analyzes some popular lossy compression methods from an image processing perspective, to obtain the theoretical compression rates. Section III presents the end to end architecture of our system in a typical wireless network. We show how we use RTCP to find out the packet

loss in order to get an estimate of the channel capacity. We then discuss how controlled lossy compression methods can be effectively used to account for reduced bandwidth, in order to transmit intelligible video streams. We present empirical results in Section IV to show the effectiveness of our system. We discuss briefly other work related to ours in Section V and finally conclude the paper with a summary of contributions of our work and directions to future work, in Section VI.

II. DATA COMPRESSION

Shannon’s “information theory” [4] showed that random processes such as audio and video data have an irreducible complexity, *entropy*, below which they cannot be compressed. It also gives means to compute the maximum transmission rate of communication, the *channel capacity*. The goal here is to reduce the transmission bandwidth requirement not to be greater than the channel capacity. The focus of much of the research from the perspective of information theory is on “lossless source coding” — *i.e.*, lossless compression of redundant information from the video stream data down to the inherent entropy, so as to reduce the bandwidth requirement needed to transmit the signal.

Video transmission has benefited tremendously from several decades of research on lossless compression. Absolute (zero) lossless compression and decompression cannot be guaranteed for video streaming — *e.g.*, network congestion or delay can result in dropped frames. For wireless networks in particular, the available channel bandwidth varies with time that can potentially result in lost packets or packets arriving much later than needed, resulting in garbled images. That is, the source coding is “virtually” lossy to the receiver, resulting in a “compressed video stream”. Fortunately, much like voice, humans can tolerate video that has virtually gone through such lossy compressions, to some extent. However, the degree of tolerance depends on how the lossy compression manifests itself. Virtual lossy compression resulting from signal distortion as it travels through the network is unpredictable and cannot be tolerated in many cases. Figure 1 shows some images that can correspond to any one frame of some video data. Figure 1(a) shows an image with no data loss, while figure 1(b) shows the same image with 33% random noise, possibly resulting from a virtual lossy compression. Obviously, larger the noise, more difficult is the frame to comprehend.

Not all lossy compressions are equal. Controlled lossy compression of the video data prior to transmitting it through the network can be effectively used to account for limited channel bandwidths. For example, greyscale transformation or size reduction that are essentially controlled lossy compressions can still be highly intelligible and informative to the recipients (as in the case of remote surveillance).

In this paper, we decouple the problems of lossless compression and controlled lossy compression and focus only on the latter. As the name implies, controlled lossy compressions reduce the overall size of the source, as a result of controlled (intentional) and lossy compression techniques. Reduction in size, greyscale conversion, reduced number of frames per second (fps), etc. are all examples of this techniques and are being effectively used in many video streaming applications. These methods reduce the overall entropy of the source and hence can

be used to transmit the source data over a (wireless) channel with a relatively smaller capacity. Though our work is aimed at reducing the source entropy of video data, we *do not* attempt to compute the actual entropy of the source (video).

A. Lossy compression methods

In this section, we discuss some popular lossy compression methods. Video streams are nothing but continuous frames of individual images displayed at a rate of about 30 per second. Of course, contiguous image frames need to be similar to each other (*i.e.*, belong to the same video “event”) for the video to be meaningful. A typical video encoder like MPEG-7 uses the difference among successive frames of the video stream to encode the data (as opposed to encoding the data of every individual frame). Any image frame is a collection of pixels each of which has a particular color. Each pixel in a color image is typically associated with a $\langle R, G, B \rangle$ triple, representing the color intensities of red, green and blue, respectively. On the other hand, each pixel in a greyscale image can be represented by a single greyscale intensity value.

We discuss below some popular lossy compression methods in the context of simple image processing techniques to obtain the theoretical compression rates. These methods can be effectively used to reduce the entropy of every individual image frame, thereby reducing its size, but still producing an intelligible image. Note that the list of transformations we discuss in this paper is by no means exhaustive — there are many other compression methods that are possible.

Color to greyscale transformation: This is a simple transformation where the color $\langle R, G, B \rangle$ of every pixel P is changed to a greyscale intensity value in the corresponding pixel P' . This can be done in many ways — the simplest being taking the mean, $\frac{(R+G+B)}{3}$, as the greyscale intensity of P' . The size of the transformed image remains the same. Since three colors are converted to a single greyscale value, the size reduction in this transformation is approximately 66.67%.

Size reduction: In this transformation the size of every individual image frame is reduced from $M \times N$ pixels to $M' \times N'$ pixels such that $M' < M$ and $N' < N$. This reduces the size of the transformed image by a factor of about $1 - \frac{M'N'}{MN}$. To ensure that the image is not stretched along any one axis, typically $\frac{M'}{M} = \frac{N'}{N}$.

Lower frame rate: As mentioned earlier in this section, video streams are contiguous image frames of an event transmitted at about 30 per second. It is possible to transmit less than 30 fps (say, up to 15 fps) and still have a reasonably good video picture. The human eye, though sensitive to lower frame rate, is capable of comprehending the picture by putting up the frames together to certain extent.

Combination of the above techniques: The preceding techniques can be combined in any order — like a greyscale image reduced in size and possibly transmitted at a lower (say, 25 instead of 30 fps) frame rate, to lead to an overall reduction in size of the video stream.

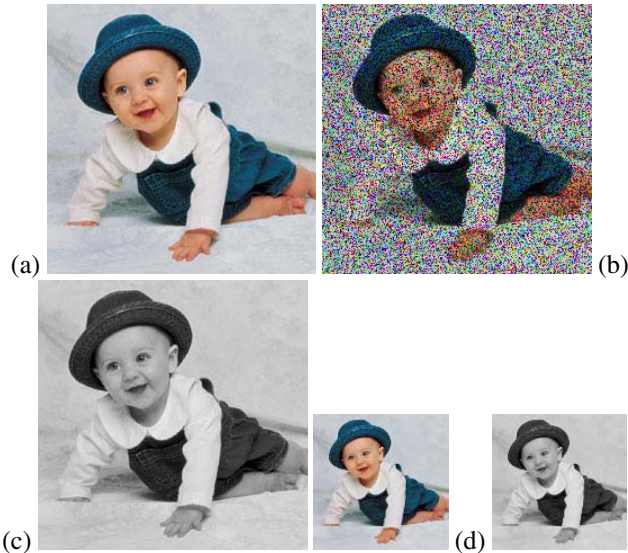


Fig. 1. Top Row: (a) Original image (b) 33% noise indicating packet loss. Bottom Row: Controlled data compression: (c) greyscale conversion (66.67%) (d) size reduction (75%) (e) size reduction and greyscale conversion (91.67%)

Compression Type	Approximate data reduction
Original color image, 30 fps	0.00%
Greyscale	66.67%
Size reduction	75.00%
Greyscale and size reduction	91.67%
Greyscale, size reduction and 22.5 fps	94.08%

TABLE I

THEORETICAL COMPRESSION RATE PER IMAGE FRAME FOR DIFFERENT COMPRESSION METHODS.

B. Example

Figure 1(a) shows a color image (that is perhaps one frame of a video stream) and also many of its transformed images. Figure 1(b) shows the same image with 33% of noise to denote a typical image frame that has undergone virtual lossy compression (say, due to packet loss). Figures 1(c), 1(d) and 1(e) show the same image with greyscale conversion, size reduction and greyscale conversion with size reduction, respectively. Greyscale conversion reduces the size of the image by 66.67% while size reduction by half along each direction reduces the overall size by 75%. A combination of these two methods yields an image frame that is one fourth in size and stores only one greyscale intensity value (as opposed to three color intensity values), thereby reducing the overall size by $1 - 0.25 \times \frac{1}{3} = 91.67\%$. Further, a controlled lossy compression method where the individual frames of a video stream are reduced in size by half, converted to greyscale and transmitted at about 22.5 fps (as opposed to the original 30 fps) saves about 94.08% of the original data! The amount of data compression that is achieved for each of these transformations is shown in Table I.

Note that these reductions are theoretical estimates for the corresponding image transformations for *one* individual image frame. The transformed image frames undergo further compression by the video encoder, before it is transmitted from the

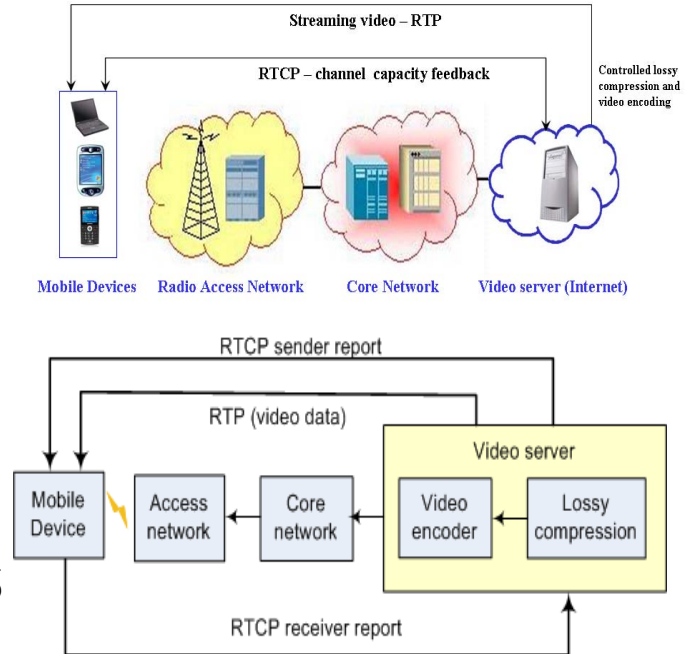


Fig. 2. (a) Top: Video transmission over a typical wireless network (b) Bottom: Video data transfer on RTP after lossy compression; RTCP reports for channel capacity estimation.

video server to the client. So the actual data that is transmitted will be different, and in fact is likely to be lower than the estimates presented in Table I.

The image frames shown in Figures 1(c), 1(d) and 1(e) are more intelligible, need less bandwidth and serve the purpose better for many applications than the image frame shown Figure 1(b), resulting from uncontrolled packet losses. Clearly, these techniques are very useful to decide dynamically the kind of video data to transmit, based on the available bandwidth. The challenge is to convey the channel capacity information to the video server, so that it can use a compression method to account for limited bandwidth.

III. ARCHITECTURE

Figure 2(a) presents a typical wireless network. It shows a cellular network like a 2G or 3G network. However, it can be a different network like WiFi. In fact, the ideas presented in this paper are true even if the access network is wired, like Ethernet. The figure shows a video server that streams video data to a wireless client. At a very high level, the end to end scenario of video data transmission consists of the following steps — capturing video data, encoding using a video encoder, streaming over a wireless network, decoding and presentation to the end user on a mobile device, as shown in Figure 2(a).

We will introduce two additional steps along this path. The first is a channel capacity feedback step to the video server, that uses RTCP [5]. We present the details of this step later in this section. The second step is a controlled data compression step that generates the video data in the required format, using one of the lossy compression methods described in Section II A. That is, using the channel capacity feedback from RTCP reports (explained later), the video server uses a lossy compression method

(or none, if there is enough bandwidth) before the data is fed to the video encoder and transmitted to the client. Figure 2(b) shows how these two steps fit in the overall architecture of video transmission. We now present an overview of RTCP and discuss how we use it for channel capacity feedback.

A. Overview of RTCP

Video streaming requests can be initiated using protocols like HTTP or SIP. The video data is typically streamed using RTP. Transmitting video data using RTP is a common practice and has been prevalent for quite sometime now. RTP defines a control protocol, RTCP, that can be used for QoS monitoring, intermedia synchronization (e.g., lip-syncing of audio and video data) using wall clock time and a corresponding RTP timestamp, source (RTP) description and identification using textual information (like unique name of the session participant, email address, telephone number), session size estimation (in terms of the number of participants) and scaling to sessions with a large number of participants (several hundreds of members).

The information among different participants of a session is carried in reports. Any participant sending data sends “sender report” (SR) to other participants receiving the data and any participant receiving data sends a “receiver report” (RR) to all the senders sending it the data. RTCP sender reports and receiver reports are similar in format, except for a 20 byte sender information that is present only in the SR, that includes the active sender information. The sender information includes timestamps of the RTP data and packet count, among other fields. The report blocks (present in both SR and RR) contain information like fraction of packets lost since the last SR was sent, cumulative number of packets lost, interarrival jitter, etc. that is useful to the sender to identify the packet loss and hence get an estimate of the bandwidth availability. SR and RR are exchanged periodically between senders and receivers for feedback control. [5] defines RTCP and gives all the details.

B. Use of RTCP for channel capacity estimation

Here, we discuss how we use RTCP for channel capacity feedback. We use RTCP to convey channel capacity feedback to the server. More precisely, the server sends the SR packet to the client periodically and the client responds with a RR. The server uses the information in the RR (timestamp, packet loss, cumulative packet loss, jitter, etc.) to get an estimate of the available bandwidth and modifies the content as needed. The frequency at which the reports are sent depends on the number of clients and servers that are active in the network — [5] recommends that RTCP reports should not take more 5% of the total bandwidth. So, the actual time interval varies with the number of participants. An important point to note is that the video server should not change its compression method frequently, based on temporary network congestions. To address this issue, the video server should take into account the cumulative packet losses (α) over a reasonable interval of time (say, several seconds to a few minutes) before it decides to change its compression method. Any meaningful mapping from α to the lossy compression method (β) of the type — if $v_1 \leq \alpha < v_2$ then use compression method β_i , can be used here. Here v_1 and v_2 are two pre-defined values. We choose the following algorithm: Use frame reduction to 20

Compression Type	Size Reduction
Original image	0.00%
Reduced frame rate (15 fps)	67.18%
Greyscale	70.0%
Size reduction	80.5%
Greyscale and size reduction	92.0%

TABLE II
COMPRESSION FOR THE VIDEO DATA FOR DIFFERENT LOSSY
COMPRESSION METHODS

fps if $0.3 \leq \alpha < 0.2$, use size reduction of 50% along each axis if $0.4 \leq \alpha < 0.3$, use frame reduction to 20 fps and size reduction if $0.9 \leq \alpha < 0.4$ and use frame reduction to 20 fps, size reduction and grey scale conversion if $\alpha > 0.9$. Obviously, the algorithm and the values specified here can be different.

IV. EXPERIMENTAL RESULTS

In this section, we describe our experimental setup and find out the compression rates for the various methods that we listed in Section II A.

A. Experimental Setup

We use an experimental setup similar to the one shown in Figure 2. We implemented a system that can send end to end live video data using a digital camera attached to a video server, using H.263 video codec. Our software is such that the picture size, video frame rate and color quality (color or greyscale) are variable parameters. That is, the camera captures the video data in real time using these parameters and they can be changed as required. We implemented our client software on two mobile platforms — laptops and Pocket-PCs. Our client is an advanced voice over IP (VoIP) SIP client that can also be used for live video data transmission. It sets up data sessions (voice or video) using SIP signaling. Efforts are underway to implement RTCP reports on the client, as described in Section III B. Our software is access technology independent, i.e., they can work on any (wired or wireless) access networks.

B. Compressions achieved

We captured 10 video clips, each running for about 2 minutes, at 30 fps. We converted each of these video clips manually *offline* (i.e., without RTCP reports’ feedback) to find out the actual compression we can achieve, for each of the techniques given in Section II A. That is, we converted each frame (after capturing the data from the camera) using one of these three compression methods and then encoded it using H.263. We performed three lossy compressions — (i) lower frame rate of 15 fps. (ii) conversion to greyscale (iii) size reduction by half along each direction (iv) conversion to greyscale and size reduction by half along each direction. Table II shows the compression that we achieved. Note that these values are different from those shown in Table I, because the values shown there are the theoretical values for one picture frame, *not* the entire video stream, after it is passed through a codec.

V. RELATED WORK

Data compression is an active field of research, and is obviously not limited to streaming video data. There has been a lot

of work in lossy compression techniques and some applications indeed use them. JPEG uses lossy compression techniques, but still provides color images of high quality color (and greyscale) images. MP3 is another application that uses lossy compression techniques to encode voice (music) with no apparent loss in quality. Cherriman and Hanzo [2] developed a H.263 based videophone system that works over mobile channels. The system uses multiple modulation schemes and uses packet dropping to reduce delay when the channel quality is poor. Whybray and Ellis [8] present an overview of ITU-T Recommendation H.263. In particular, the operation and impact on quality and bit-rate of the four H.263 Annexes (D, E, F and G) which specify optional enhanced modes of operation are discussed, with supporting results. Unlike all the approaches listed above, we use controlled lossy compression techniques to reduce the inherent entropy of the source to make up for limited bandwidth.

Busse *et al.* [1] use RTCP to dynamically adjust the bandwidth requirements of multimedia applications. They use RTCP reports from multiple receivers and use them to decrease, or hold or increase the bandwidth of the multimedia sessions. Our approach is similar to theirs in using RTCP reports for feedback. However, unlike their approach, we use lossy compressions and modify the video source content. Wong *et al.* [9] and many other references listed in that work use rate adaptation to address the QoS issues. This method typically estimates the channel quality and adjusts the transmission rate accordingly. This method is restricted to over the air channel capacity and does not address the end to end QoS issues. Our approach can also be considered as a rate adaptation method that uses RTCP. In fact, it can be used in *addition* to the various other rate adaptation methods that are available today. Siddiqui *et al.* [3], Van den Berg *et al.* [7] and the various other references listed in them address the QoS issues in wired and wireless networks. Shi *et al.* [6] present a new periodic broadcast protocol that can be optimized for clients with heterogeneous reception bandwidth and QoS requirements. In contrast to the all the above listed methods, we estimate channel capacity using RTCP reports and use them to modify the source data content using popular lossy compression methods. There is plenty of other work related to data compression, channel capacity estimation and QoS related issues. We can use many of those compression methods and QoS methods (e.g., reserving resources) in addition to our approach of source data modification.

VI. CONCLUSIONS

We present a summary of the contributions of our work here and give directions to future work.

A. Contributions

This paper addresses an important problem in today's wireless networks — limited bandwidth over the air interface. This, coupled with difficulties associated in assuring end to end QoS guarantees, poses a serious problem to data intense real time applications like video streaming. We discuss the theoretical compression rates that can be achieved for each image frame, using some of the popular lossy compression methods. We show that controlled lossy methods can be used to achieve high compression rates (of over 90%) and still result in intelligible video

streams. Further, we argue that in many situations (like remote surveillance) controlled compressions are a better alternative to packet losses that result in jittery video streams.

We then present an end to end architecture where a video server receives feedback from mobile clients about packet losses using RTCP, thus gaining an estimate of the available bandwidth. If the bandwidth is limited, the video server applies the lossy compression method to produce a stream that needs a lower bandwidth (than the original stream) and still presents an intelligible stream to the user. We present empirical results to compare the actual compression rates that can be achieved for video data to the theoretical estimates.

B. Future Work

There are several possible extensions to this work. Some of them are pretty straightforward like finding more image transformation methods and better compression schemes (be it lossy or lossless). In our work, we used a static sequence of mapping based on the packet loss to the lossy compression method, without any consideration to user preferences. The video server can use more sophisticated methods to provide the best possible video stream, without compromising too much on the quality (in terms of size, frame rate, etc.) on the video stream, taking into considerations the clients' preferences and their requirements.

We did not address the scalability issues of our approach. That is, there can be instances where video data is broadcast to several hundreds or thousands of users. RTCP addresses this issue and defines guidelines to ensure that the network is not flooded with RTCP reports. An interesting extension would be how to incorporate these guidelines into our work to scale it up to cases where many users are involved.

Yet another extension to prioritize different users — e.g., based on the kind of service they subscribed to. When multiple users are competing for the bandwidth, a user with a premium subscription needs to get be given preference in such a way that his video stream is not subjected to a lossy compression, or it is compressed with minimal data loss compared to other users.

REFERENCES

- [1] I. Busse, B. Deffner and H. Schulzrinne, "Dynamic QoS control of multimedia applications based on RTP", *Computer Communications*, Jan. 1996.
- [2] P. Cherriman and L. Hanzo, "Programmable H.263-based Wireless Video Transceivers for Interference-limited Environments", *IEEE Transactions on Circuits and Systems for Video Technology Journal*, Jun 1998
- [3] M.A. Siddiqui, K. Guo, S. Rangarajan and S. Paul, "End-to-end QoS support for SIP sessions in CDMA2000 networks", *Bell Labs Technical Journal* 9(3): 135-153 (2004)
- [4] C.E. Shannon, "A mathematical theory of communication", *Bell Systems Technical Journal*, 27: 379 – 423, 623 – 656, 1948
- [5] H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson, "RFC 3550 - RTP: A Transport Protocol for Real-Time Applications", Jul 2003
- [6] L. Shi, P. Sessini, A. Mahanti, Z. Li and D. L. Eager, "Scalable streaming for heterogeneous clients", *ACM Multimedia 2006*: 337-346
- [7] E. Van den Berg, S. Madhani and T. Zhang, "Real-Time Comparison of Quality of Interfaces by Mobile Devices over Heterogeneous Radio Networks", *First International Conference on QoS in Heterogeneous Wired/Wireless Networks*, Oct 2004
- [8] M.W. Whybray and W. Ellis, "H.263-video coding recommendation for PSTN videophone and multimedia" *IEEE Colloquium on Low Bit Image Coding*, London, Jun 1995
- [9] S. H. Y. Wong, H. Yang, S. Lu and V. Bharghavan, "Robust rate adaptation for 802.11 wireless networks", *International Conference on Mobile Computing and Networking*, 2006.