Video Streaming over Bluetooth

Davide Catania
Department of Communications
and Computer Engineering
University of Malta
Email: davidecatania@gmail.com

Saviour Zammit
Department of Communications
and Computer Engineering
University of Malta
Email: saviour.zammit@um.edu.mt

Abstract

In recent years, multimedia content has become more accessible to mobile phone devices increasing the demand for multimedia services. Streaming video to or from mobile phones over mobile phone operator networks is one option. In this paper we report on the result of a study which analyzes the suitability of using the Bluetooth network as a last hop network for streaming video to and from mobile phone devices [1]. A number of studies have been reported in the literature, simulating video streaming over Bluetooth. However, few field studies have been reported fuelling the need to build an implementation infrastructure to conduct an empirical study using mobile phone devices, in video streaming applications. In our study we have implemented a testbed comprising a Linux-based Bluetooth video-streaming gateway and a Nokia mobile phone device to stream video clips and real-time video to and from the mobile phone over a Bluetooth connection, using both pre-recorded video and real-time streams from the mobile phone's onboard video camera. The testbed allows various Bluetooth network protocols and parameters to be tested in our framework. The work carried out reinforces the importance of adequate packetization, which proved to be beneficial even with higher protocol layers such as the L2CAP protocol. The data throughputs achieved using Bluetooth v1.1 and Bluetooth v2.0 adapters were also compared and the effect of Wi-Fi interference proved to be detrimental to the performance of the Bluetooth network's data throughput. The use of L2CAP and RFCOMM sockets were compared, highlighting the importance of the choice of an adequate protocol. Video quality degradation at different distances when transferring video over Bluetooth was measured in terms of the mean square error metric. It was shown that the mobile phone device is indeed a resource constrained device and special care must be taken to ensure working streaming-video solutions over Bluetooth.

Index Terms

Bluetooth, mobile, multimedia, video, streaming

1. Introduction

Bluetooth is a wireless technology operating in the unlicensed 2.4GHz Industrial, Scientific, and Medical (ISM) band. The current version of Bluetooth, v2.0, defines two main modulation schemes. These are the Basic Rate, which uses Gaussian Frequency Shift Keying (GFSK) to maintain compatibility with the older Bluetooth v1.1 specification, and the Enhanced Data Rate which uses two variants of PSK, $\frac{\pi}{4}$-DQPSK and 8DPSK. The symbol rate for all schemes is 1Msym/s with the Basic Rate giving a gross data rate of 1Mbps, 2Mbps for the Enhanced Data rate using $\frac{\pi}{4}$-DQPSK and 3Mbps for the Enhanced Data rate using 8DPSK [2]. Given these data throughputs and the wide availability of Bluetooth radio on mobile devices, it becomes increasingly appealing to study whether Bluetooth technology is suitable to carry video streams in a last hop network scenario. In this study we provide a concrete working framework showing how video streaming over Bluetooth can be implemented between a mobile phone device and a desktop workstation. The choice of peripherals, that is, the mobile phone device and desktop workstation are chosen strategically and illustrate a practical case. In recent years mobile phone devices, started to incorporate more multimedia functionalities. However due to the current network infrastructure and other reasons, the cost associated with using such multimedia services, as for example streaming video from or to the mobile phone device, is still very high.

A good number of studies simulating video streaming over Bluetooth have been published in the literature [3][4]. However few implementation frameworks were encountered. There are significant limitations associated with using practical frameworks when compared to the flexibility and configurability available in simulation packages. The aim of this paper is to highlight these limitations in relation to the testbed used and to point out the usefulness of the test results that were obtained.

2. Theory and Previous Work

The Bluetooth specification offers a number of packet types which one can use when transferring data on the baseband layer. A list of packet types available, along with some associated details, including data throughput is listed in Table 1.

Table 1. Summary of the ACL packet types available for use in Bluetooth v2.0

Type	Payload Header (bytes)	User Payload (bytes)	FEC	Forward Assymetric Rate in kbit/s	Reverse Assymetric Rate in kbit/s
DM1	1	0-17	2/3	108.8	108.8
DH1	1	0-27	None	172.8	172.8
DM3	2	0-121	2/3	387.2	54.4
DH3	2	0-183	None	585.6	86.4
DM5	2	0-224	2/3	477.8	36.3
DH5	2	0-339	None	723.2	57.6
2-DH1	2	0-54	None	345.6	345.6
2-DH3	2	0-367	None	1174.4	172.8
2-DH5	2	0-679	None	1448.5	115.2
3-DH1	2	0-83	None	531.2	531.2
3-DH3	2	0-552	None	1766.4	235.6
3-DH5	2	0-1021	None	2178.1	177.1

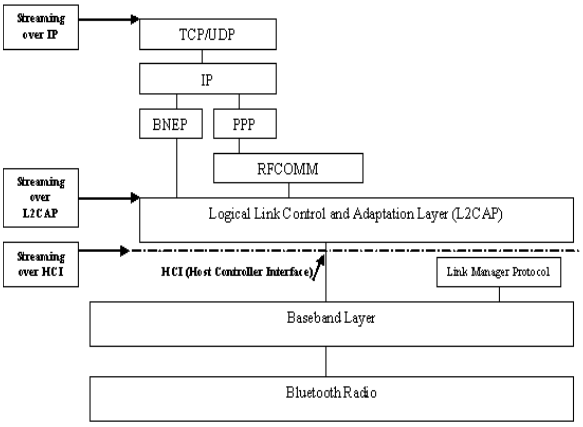


Figure 1. Simplified version of the Bluetooth protocol stack, highlighting the main choices proposed for video streaming

Furthermore Bluetooth supports two link types, the Synchronous Connection Oriented (SCO) link and the Asynchronous Connectionless link (ACL). SCO links are generally used for voice applications. Data is never retransmitted when using SCO links. ACL links on the other hand provide much more flexibility and configurability. ACL can be used for both asymmetric and symmetric data conditions. This type of link can give forward data rates of up to 723.2kbit/s using the basic data rate and up to 2178.1kbit/s using the enhanced data rate. For video streaming over Bluetooth we used ACL links to transfer our data since this type of link provides increased data throughput, flexibility, and added configurability when it comes to protecting packets over the wireless Bluetooth network. These packets are, however, only available at the baseband layer. In [3], Xiaohang gives a brief overview of the options available for video streaming over Bluetooth and highlights the main issues associated with video streaming over Bluetooth.

The Bluetooth specification provides a protocol stack which enables interconnectivity between Bluetooth devices, legacy devices and other systems. The Bluetooth protocol stack is shown in Figure 1. As outlined by Xiaohang in [3], video streaming over Bluetooth can use one of the three options listed in Figure 1 that is streaming over HCI, over L2CAP or over IP. At a first glance, streaming over HCI can seem to be the most viable solution due to the lowest overhead involved. However attempts reported in [4] by Chia et al. have shown that in practice, streaming over HCI can be quite problematic. For instance most Bluetooth devices expect an L2CAP header after an HCI packet, leading to discarded packets if the L2CAP header is not present. Also, the sizes of packets used when streaming over HCI are device dependent since most devices have different buffer sizes. The HCI layer also lacks segmentation and reassembly support which is available in higher layers such as L2CAP. These difficulties increase the implementation complexity required and make video streaming over HCI problematic. The second option proposed is streaming over L2CAP. The L2CAP protocol is also a lower layer protocol and although there is more overhead involved than in the HCI option, the overhead introduced remains relatively low. Moreover the implementation complexity involved is much smaller than that in HCI. The third option proposed is streaming over IP. Streaming over Bluetooth's IP adaptation could be considered as the easiest way of streaming video over

Bluetooth since currently deployed video streaming over IP technologies are widely available. However this method produces significant overhead, which can pose considerable problems for networks with limited bandwidth such as Bluetooth. Apart from considering the disadvantages and advantages identified here, one must also consider the options available in a realisable implementation framework.

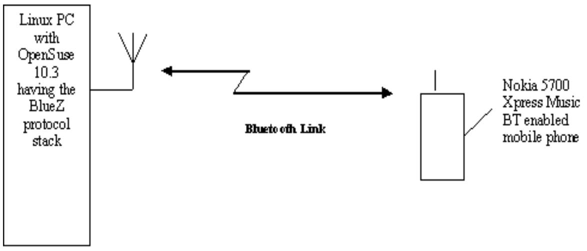

Linux PC with OpenSuse 10.3 having the BlueZ protocol stack
Bluetooth Link
Nokia 5700 Xpress Music BT enabled mobile phone

Figure 2. System Overview

3. Implementation Framework

Our testbed mainly consisted of a Nokia S60 3rd Edition Feature Pack 1 mobile phone device, specifically the Nokia 5700 Xpress [5], a Bluetooth v1.1 USB dongle [6] and a Bluetooth v2.0 dongle [7] attached to a Linux desktop workstation. The choice of using a Nokia mobile phone is justified by the wide availability of support by Symbian and Nokia when it comes to mobile phone programming. In addition to this Nokia also provides several free tools such as the basic version of the Carbide.c++ IDE and the Nokia S60 SDK which can be used to develop mobile phone applications. It is also worth mentioning that one can write mobile phone software in a variety of languages such as Symbian C++ (a variant of C++), Java J2ME, Python and Flash. Since the Symbian operating system, the operating system running on all S60 Nokia mobile phones, was written in C++, our programs were developed in Symbian C++ in an attempt to improve the speed of operation. Also, most low level API's are only available for Symbian C++ programmers, making future development of the software easier and more flexible. For the desktop workstation, Linux was chosen as the underlying operating system due to the availability of the BlueZ protocol stack [8]. In contrast with other platforms, the BlueZ protocol stack offers support for transferring packets using HCI, L2CAP and RFCOMM protocols [9]. On the other hand current Microsoft Bluetooth API's only support the RFCOMM protocol, making a Windows based system less suitable for our study. Although the Symbian framework is also quite flexible, there were various limitations in using some protocols and due to a missing PAN profile implementation in the Nokia S60 SDK [10], we could not implement video streaming over IP implementation, limiting our solution to using the L2CAP and RFCOMM protocols for transferring video over the Bluetooth network. Figure 2 shows an overview of our system.

After having determined the hardware details of our implementation, the software which enabled Bluetooth connectivity between the mobile phone device and the Bluetooth enabled Linux desktop workstation was developed and three main modes of operation were established.

In the first mode of operation offline video streaming over Bluetooth was implemented and the Linux workstation could transmit a pre-encoded video stream, using fixed packet sizes disregarding video slice boundaries, to the mobile phone device. As soon as the mobile phone device received a suitable amount of data, the mobile phone's video player, accessed through the CVideoPlayerUtility API [11], was launched automatically, and playback of the video currently being received was initiated. Ideally one would decode the video frames manually using the onboard mobile phone video decoder (a software decoder could also be used but this is not desirable since significant processing power would be utilized unnecessarily). However, the SDK for S60 3rd Edition Feature Pack 1 devices does not provide access to the DevVideo API, which decodes video frames using the mobile phone's onboard hardware decoder (this API is available for S60 3rd Edition Feature Pack 2 devices with limited documentation) and hence video frames could not be manually decoded. Also the CVideoPlayerUtility API can only open video streams which are located in a video container file. To initiate playback prior to complete reception of the video file, the sample table-chunk-offset atom of the video stream had to be moved to the front of the video file. Here video packets were successfully transmitted using both the L2CAP and RFCOMM protocols. It was also noted that in some cases, very small packet sizes of around 700 bytes and less, lead to video stutter at a later stage of the video playback due to incorrect file writing operations on the mobile phone device.

These problems were not encountered in the second mode of operation where a pre-encoded video stream was transmitted from the mobile phone device to the Linux desktop workstation, confirming that the mobile phone device is indeed limited in processing power when compared to a normal desktop workstation.

In the third and last mode of operation, a realtime H263 encoded video stream from the mobile phone's onboard video camera was transmitted via Bluetooth to the Linux desktop workstation. On the Linux desktop workstation video frames were decoded by the VLC media player [12], which was automatically launched after having received a suitable number of packets [1].

4. Results and Analysis

After having successfully demonstrated the three modes of operation, the second mode of operation (mobile phone sending a pre-encoded video stream to Linux station) and the third mode of operation (mobile phone's onboard video camera sending a realtime H263 encoded video stream to Linux station) were subjected to further tests. The results obtained when testing the second mode of operation are reported first, followed by the results obtained testing the third mode of operation.

In the first test, using transmission mode two, the data throughput, mean interarrival time, and inter-packet delay variation of the Bluetooth network was tested using different fixed packet sizes when transmitting the video stream over the L2CAP layer. This test was inspired by previous work carried out in [13] by Razavi et al., showing the importance of using DH5, 2DH5, or 3DH5 baseband packets when transferring video over Bluetooth in order to maximize data throughput and effectively improve video quality. Although our video stream was not transferred over the baseband layer, but using the L2CAP protocol, it was also found that larger packet sizes increase the data throughput as shown graphically in figure 3. The packet sizes were chosen strategically, considering the L2CAP header introduced, to induce the Bluetooth controller to use certain baseband packet types at the L2CAP layer. In most cases, the data throughput obtained matched with the theoretical data throughput possible using a particular packet size although in some rare cases the Bluetooth controller promoted the packets to higher data throughput baseband packets. It must also be noted, as it can be clearly seen from figure 3, that although in general larger packet sizes increase the data throughput considerably, they also negatively increase the mean inter-arrival time.

In another test, the data throughput when using L2CAP and RFCOMM sockets was compared and it was established that the overhead introduced by the RFCOMM protocol reduces the data throughput, making the L2CAP protocol more suitable for high quality video streaming over Bluetooth as it can be observed from figure 4.

After establishing the importance of choosing an appropriate packet size and protocol for transferring data

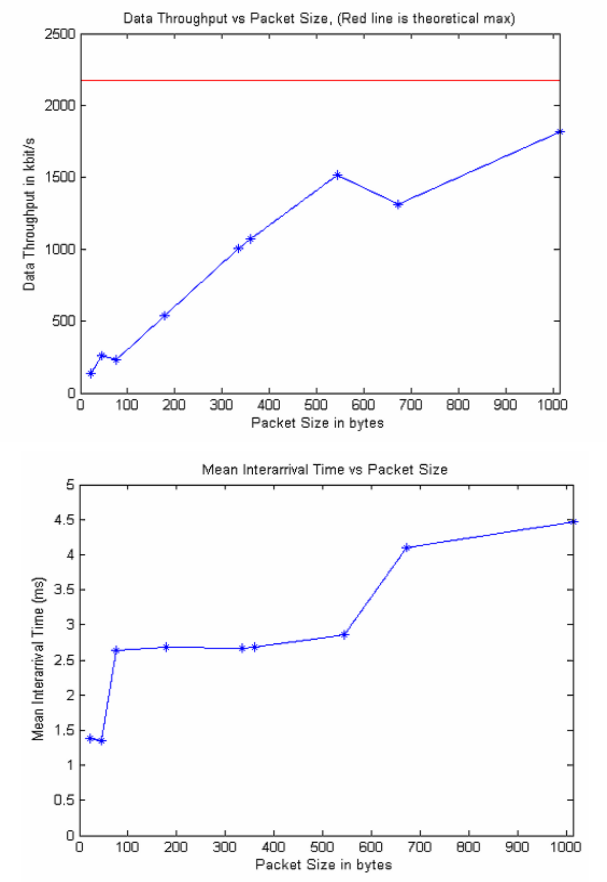
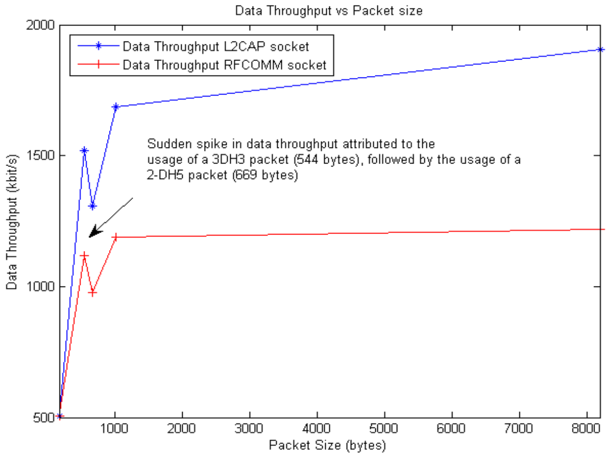


Figure 3. Average Data Throughput vs. Packet Size (top), Mean Interarrival Time vs. Packet Size (bottom)



Sudden spike in data throughput attributed to the usage of a 3DH3 packet (544 bytes), followed by the usage of a 2-DH5 packet (669 bytes)

Figure 4. Average Data Throughput vs. Packet Size comparison using RFCOMM and L2CAP sockets

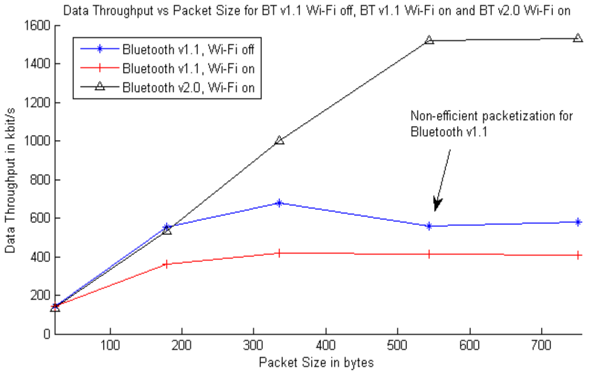


Figure 5. Bluetooth v1.1 vs. Bluetooth v2.0 Data Rates using different packet sizes considering Wi-Fi interference

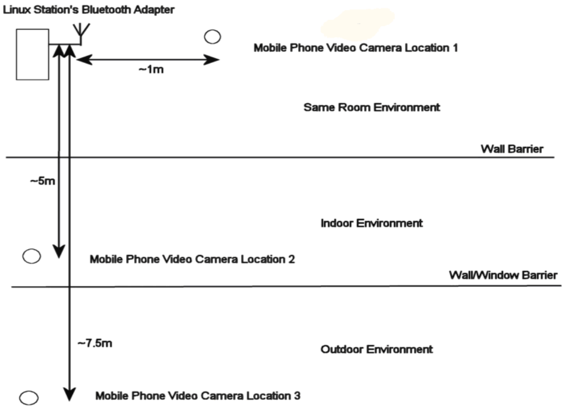

Linux Station's Bluetooth Adapter
~1m
Mobile Phone Video Camera Location 1
Same Room Environment
Wall Barrier
~5m
Indoor Environment
Mobile Phone Video Camera Location 2
Wall/Window Barrier
~7.5m
Outdoor Environment
Mobile Phone Video Camera Location 3

Figure 6. Testing Environment for third mode of operation

over Bluetooth, the data throughputs of Bluetooth v1.1 and Bluetooth v2.0 were compared. With most legacy mobile phone devices equipped with a Bluetooth v1.1 radio chip it is important to analyze the possibilities and limitations of this version. The effect of Wi-Fi interference which also operates in the 2.4GHz band was also investigated, and it was proven to be considerably detrimental to the data throughput of Bluetooth v1.1, which does not support adaptive frequency hopping unlike the newer version of Bluetooth. In adaptive frequency hopping, the system will try to sense the frequencies being commonly used and will then try to avoid them, hence reducing interference, which will in turn reduce packet repeat requests, hence increasing the data throughput. Figure 5 shows the data throughput achievable with Bluetooth v1.1 and Bluetooth v2.0, and also highlights the negative effect of Wi-Fi interference on Bluetooth v1.1's data throughput.

In the third mode of operation, being a real time system, the L2CAP socket was set to operate in unreliable mode, unlike previous modes of operation. This means that if a packet has not been successfully transferred to the receiving end, within a certain timeout, the packet will be dropped. A realtime video stream having 15 frames per second, with frame size 176x144, and an average data rate of 64kbit/s was transmitted from the mobile phone's video camera to the desktop workstation via Bluetooth using unreliable L2CAP, at various distances separating the sending and receiving nodes. The majority of mobile phones contain a class 2 Bluetooth radio chip which limits the working distance to a maximum range of 10m. Hence three distances of 1m, 5m and 7.5m were tested strategically in three different environments as depicted in figure 6. Two thousand H263 encoded video frames were transmitted from the mobile phone device, and by comparing the original video stream with the received stream, the video quality degradation at various distances was assessed using the mean square error metric.

As depicted in figure 7, the further the distance, the more video frames are dropped and hence the worse the video quality degradation. The lost video frames are indicated in figure 7 by pink peaks saturating at a MSE of 2000. The video quality degradation does not only depend on the number of frames being dropped but also on the type of video frame being dropped. Usually video streams are composed of I-frames which are extremely important in reconstructing a picture, and P-frames which are less important. Hence, if an I-frame is lost the mean square error of the received video will be larger than if a P-frame is lost.

5. Conclusion

This study confirms that video streaming over Bluetooth is indeed realisable, although further video decoding tools are desirable, especially on the mobile phone platform. The L2CAP protocol was found to be the most suitable protocol to transfer video over Bluetooth, given the implementation tools available and the added configurability it provides when compared to higher layers. The packet size used when transferring data over Bluetooth was found to be extremely important for increasing the data throughput. The study also confirmed that the adaptive frequency hopping scheme available in Bluetooth v2.0 is quite effective in combating Wi-Fi interference.

Further work on video streaming over Bluetooth between mobile phones and desktop workstations, must include the development of software which utilizes

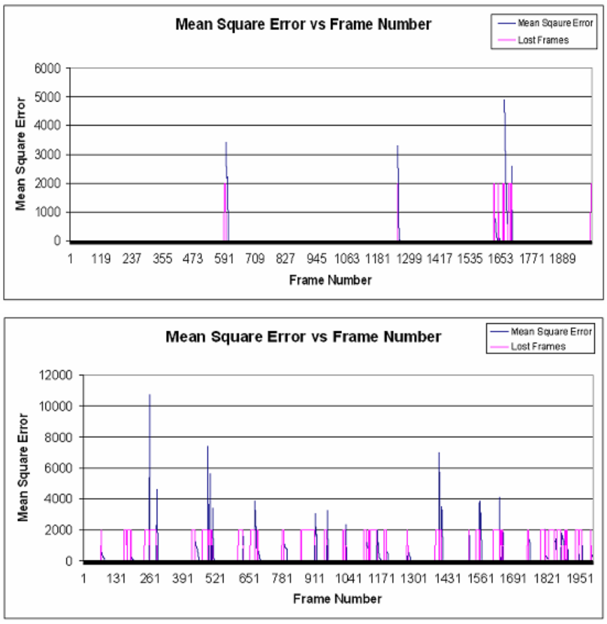


Figure 7. Mean Square Error Measurements for Location Point 2, 5m (top) and Location Point 3, 7.5m (bottom)

the mobile phone video hardware decoders available. Other optimization techniques read in literature must also be tested, in order to assess their effectiveness in actual implementations. For instance, work carried out by Kapoor et al. in [14], has produced considerable video quality improvements. In [14] the authors show that by prioritizing I-frames over P-frames using the ARQ timeout parameter, that is allowing a much larger ARQ timeout for I-frames than for P-frames, one can achieve better perceived video quality at the receiver.

References

[1] D. Catania, *"Video Streaming over Bluetooth"*, B.Eng. dissertation, University of Malta, Malta, 2008

[2] Bluetooth Special Interest Group, *"Specification of the Bluetooth System"*, v2.0 + EDR.

[3] W. Xiaohang, *"Video Streaming over Bluetooth: A Survey"*, [Online]. Available: http://www.comp.nus.edu.sg/ cs5248/0304S1/surveys /wang-bluetooth.pdf

[4] C. H. Chia, M.S. Beg, *"MPEG-4 Video Transmission over Bluetooth Links"*, in 2002 IEEE International Conference on Personal Wireless Communications, Dec. 2002, pp. 280- 284.

[5] Forum Nokia, *Device Details Nokia 5700 Xpress Music.* [Online]. Available: http://www.forum.nokia.com/devices/5700_Xpress Music

[6] Fujitech, FBT-01. [Online]. Available: http://www.fujitech.com/hot_product.php

[7] Edimax, EB-DGC1. [Online]. Available: http://www.edimax.com/en/produce_detail.php?pd _id=126&pl1_id=13&pl2_id=43

[8] BlueZ, Official Linux Bluetooth Protocol Stack, [Online], Available: http://www.bluez.org/download.html

[9] A.S. Haung, L. Rudolph, *"Bluetooth Essentials for Programmers"*, New York, Cambridge University Press, 2007

[10] Iain Campbell, *"Symbian OS Communications Programming"*, West Sussex, Wiley, 2007

[11] Forum Nokia Wiki, *"How to play a video file using CVideoPlayerUtility"*, [Online] Available: http://wiki.forum.nokia.com/index.php/How_to_play _a_video_file_using_CVideoPlayerUtility

[12] VideoLAN, VLC Media Player, [Online] Available: http://www.videolan.org/vlc/

[13] R. Razavi, M. Fleury, E. Jammeh and M. Ghanbari, *"An Efficient Packetization Scheme for Bluetooth Video Transmission"*, Electronic Letters, vol. 42, no. 20, pp. 1143-1145, 2006.

[14] R. Kapoor, M. Cesana, M. Gerla, *"Link Layer Support for Streaming MPEG Video over Wireless Links"*, 2003 International Conference on Computer Communications and Networks (ICCCN 2003), 2003.