A Video Streaming Application Using Mobile Media Application Programming Interface

Ary Mazharuddin Shiddiqi, Henry Pratama, Henning Titi Ciptaningtyas
Dept. of Informatics, Fac. of Information Technology, Sepuluh Nopember Institute of Technology
Jl. Raya ITS, Kampus ITS, Sukolilo, Surabaya, Telp 031-5939214, Fax 031- 5913804
e-mail: ary.shiddiqi @cs.its.ac.id, henry_p@cs.its.ac.id, henning@its-sby.edu

Abstrak

Dewasa ini perkembangan teknologi telepon selular berkembang dengan pesat. Perkembangan ini mengarah pada lahirnya mobile multimedia phone yang mendukung koneksi Wireless Local Area Network (WLAN). Akan tetapi penggunaan teknologi WLAN pada telepon seluler untuk mengakses video secara streaming sangat jarang ditemui. Sedangkan Symbian S60 saat ini sebagai sistem operasi mobile multimedia phone sangat handal dalam menangani berbagai macam media seperti video. Pembahasan dalam penelitian ini menyajikan pembuatan aplikasi video streaming pada mobile phone melalui koneksi WLAN dengan menggunakan teknologi JSR 135 atau lebih dikenal dengan Mobile Media API (MMAPI). MMAPI digunakan untuk mengontrol proses video streaming dan fitur-fitur pendukungnya. Aplikasi ini akan menggunakan 2 pilihan protokol yaitu RTSP dan HTTP. Hasil uji coba menunjukkan bahwa penggunaan MMAPI pada telepon seluler berbasis Symbian 60 untuk melakukan video streaming dapat diterapkan dan mempunyai kehandalan yang baik. Hal ini ditunjukkan dengan nilai paket loss 0% pada koneksi yang reliable dan waktu yang dibutuhkan untuk memutar file multimedia tidak terpengaruh oleh besar file yang dibuka.

***Kata kunci**: Video Streaming, WLAN, Symbian S60, MMAPI.*

Abstract

Recently, the development of mobile phone technology is growing rapidly. These developments led to the emerging of a multimedia mobile phone that supports Wireless Local Area Network (WLAN). However, the use of WLAN technology on mobile phones to access the streaming video is very rarely employed, while the current S60 Symbian operating system as a multimedia mobile phone is very reliable in handling a video. This study presents the making of a video streaming application in mobile phone via a WLAN connection using JSR 135 technology or the Mobile Media API (MMAPI). MMAPI is used to control the process of video streaming and its features. The application uses the two protocols; RTSP and HTTP. Experiment results show that the use of MMAPI on Symbian 60 based mobile phones to do video streaming is feasible and has a good reliability as 0% packet loss on connection. In addition, the times required to play multimedia files are not affected by the size of video streaming files.

***Key Words:** Streaming Video, WLAN, Symbian S60, MMAPI.*

1. Introduction

In this modern era, mobile phone technology has been growing rapidly. The current trend of the development is on multimedia mobile phones. Most mobile phone manufacturers use Symbian operating system for their products. The latest Symbian version of the multimedia mobile phones is Symbian version 60. The Symbian 60 has the editions according to features being added on starting from S60 First Edition, S60 Second Edition, S60 Third Edition and the most recent version is S60 Fifth Edition [1], [2], [3].

As the growing demand for multimedia quality, the growing in the multimedia technology is also balancing. In the S60 Third Edition and Fifth Edition, the phone has been equipped by a variety of connection options to support multimedia flexibility [2].

However, the use of streaming technology in mobile phones with the S60 operating system is very rare. Though it has been known that mobile phones with that type already has the feature of Wireless Local Area Network (WLAN). Video streaming technology enables mobile users to watch videos without having to have the video files on a cell phone or download it first from other devices. It can be imagined if the video file with the size of 100MB should be downloaded to a mobile phone, then it will take a long time to finish and a large storage. By utilizing streaming video technology on mobile phones, this problem will not happen [2]-[8].

J2ME or Java 2 Micro Edition has a mobile library media API (MMAPI) or JSR (Java Specification Requests) 135 used to handle multimedia files [9]. This study developed a video streaming application on mobile phones based on Symbian OS via WLAN using a Mobile Media API (MMAPI).

2. Research Method

This research built an application called Pocket VidStream used to play streaming videos on mobile devices. To run the video streaming, the software on a mobile device that acts as a client should connect to a server that functions as a provider of video streaming services. We used Mobile Media API (MMAPI) of Java 2 Microedition to handle data transmission protocol (protocol handling) and handling the contents of the data (content handling). The handling protocol reads data from source (such as files, streaming servers, captured device) and then processes them in the media processing system. While the content handling processes the media data (such as parsing or decoding) and then renders to output devices such as audio speakers or video displays. At the API, there are two high-level objects used; they are data source and player. Each object represents one of the multimedia processing. Data source object represents the protocol handling, while the player object represents the content handling.

In this application, video streaming uses on-demand streaming concept. The protocol used in on-demand streaming protocol is HTTP and RTSP. On-demand streaming is activated by user request and can be presented at any time in accordance with requests from the client. In other words, on-demand streaming is similar to seeing video tapes where we can pause and play the video.

The concept of client server used is as follows. Server: a computer that is used as a server streaming (RTSP servers and HTTP servers), and client: a mobile device that performs requests to a Server.

The device used to run this application has to support the necessary infrastructure and support the type of media files to be played. To play a media, it takes two objects, namely: DataSource and Player. DataSource handles details of how to obtain data from sources that are available. Source comes from servers that provide streaming service. Player need not be too concerned about where the data originated from or how to get it. The player only needs to read data from DataSource, processes, displays and plays media playback on the output device. There is a third party in this scenario, i.e. the Manager. The manager creates Player from the DataSource. It creates a Player from the media source location (URL), DataSource and InputStreams.

Figure 1 and Figure 2 show the general system of Pocket VidStream. Firstly, users must establish a connection to the WLAN network via wireless access point. Then, client applications will perform XML parsing to the server in accordance with the protocol and the type of video selected by the user. At the client, the XML parsing is done to obtain data title and the URL of the video that is stored in XML files on the server. When the XML parsing is complete, the client application will display a list of video titles available at the server which then the user can select.

In this application, users can stream both by HTTP and RTSP protocols. Both protocols have their own advantages. The HTTP protocol downloads video files at the content server (HTTP server) and then save it into a buffer before being posted, so it takes longer for large sized files. While the RTSP protocol streams the video in a real time fashion. By using RTSP protocol, when the client request to the server, the server will direct its response by sending streams of video files in sequence, and the client can immediately play the video stream.

HTTP protocol is better used if the file is small, and broadcasters wants to send a file with a higher quality of streaming media (RTSP Server). The advantage of using the HTTP

protocol is high video quality. This is because the file is completely transferred before being played. In addition, the client application is still able to watch the video because the video file has been downloaded and stored in a buffer. However, the disadvantage of using HTTP protocol is the users have to wait longer and if the connection is lost or dies the users can only play part of the video file that has been stored in the buffer.

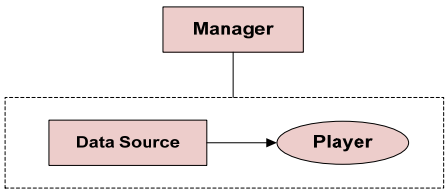


Figure 1. Object Manager interactions, DataSource and Player

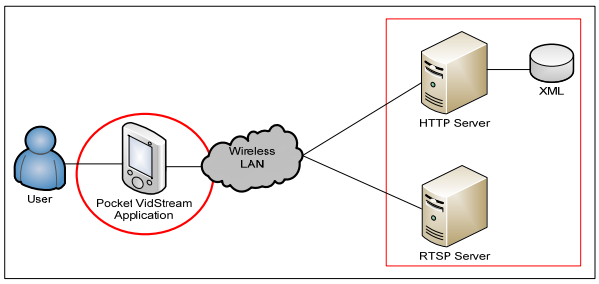


Figure 2. Object Manager interactions, DataSource and Player

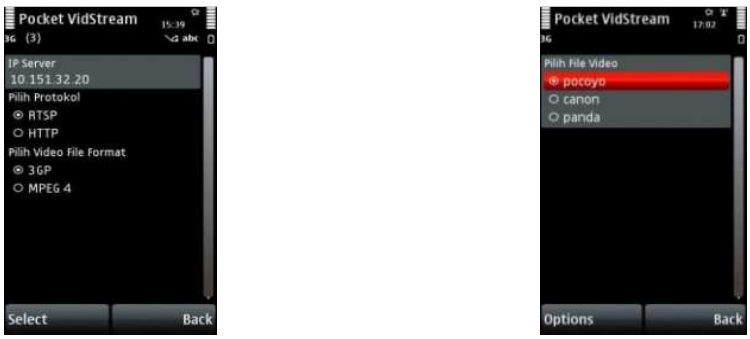
On the other hand, with the RTSP protocol, users do not have wait too long to play the video because the video playback process performed in real time while the server keeps transferring the video streaming until the whole file is completely transferred.

3. Results and Analysis

Experiments are conducted to observe the system's functionality, compatibility, and performance.

3.1. Functionality Experiment

This experiment was conducted to test the basic functions of this software to run properly. Figure 3 shows the process configuration settings to determine the server's IP address, protocol options and the type of video content that will be used. The results indicated in Figure 3a. Then, after performing the configuration process, the client will do the parsing of XML to display a list of video titles that will be selected for the video stream as shown in Figure 3b.



a. b.

Figure 3. The process configuration of the systems, (a). The configuration settings, b. Video title list

When a title is selected by user, the application streaming process will open a connection to the server. Once the connection is built, the process of transferring data between servers to the Pocket VidStream application starts as shown in Figure 4.

Figure 4. Video Streaming Processes

Figure 5. Mute Sound Experiment

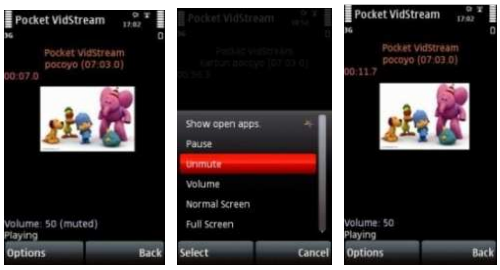


Figure 6. Unmute Sound Experiment

Figure 7. Volume Adjustment Experiment

Figure 8. Pausing Video Streaming

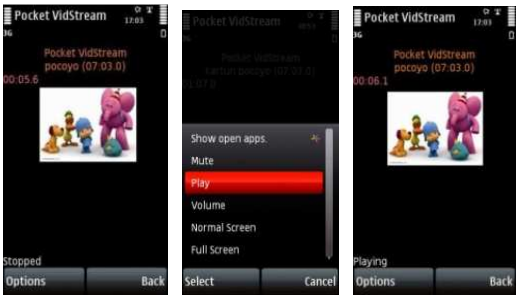


Figure 9. Playing Back Video Streaming

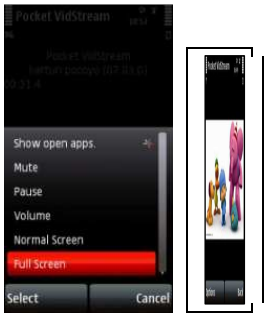


Figure 10. Full Screen Viewing

Figure 11. Normal Screen Viewing

During the transfer process of the streaming video, this application can hold the broadcast; this process is called a mute. Figure 5 shows this experiment. To restore the sound that has been done previously muted we can do unmute process as shown in Figure 6. During the delivery of streaming video is in progress, this application can perform voice settings in the menu features. This function is shown in Figure 7.

The process to hang on and resume the video streaming delivery process that is underway called the pause and resume process. The pause and resume of the video is shown in the Figure 8 and 9.

This application has a feature to display streaming video with a maximum screen size. Experiment of full-screen view is shown in Figure 10. To return to the previous viewing streaming video, users simply need to press the button on the menu screen to normal as shown Figure 11.

3.2. Compatibility Experiment

Compatibility testing is done to observe the compatibility of the system to other devices that uses the same application. This experiment also observes the maximum range of mobile devices that can be reached. Compatibility is measured by the functioning of all the features and functionality of the system. This compatibility test measured the functions of the system in running the protocol and playing video content over the WLAN connection. There are two protocols used in this experiment, i.e. HTTP and RTSP. This experiment is shown in Table 1. Table 2 shows that this system runs well on mobile phones with Symbian operating system 5th Edition. However, this application cannot run the RTSP protocol over a WLAN connection.

Table 1. Mobile Phone Specifications

	Handphone Nokia			
	5800	N97	E71	E52
MIDP	2.1	2.1	2.1	2.1
CLDC	1.1	1.1	1.1	1.1
Symbian OS	v9.4	v9.4	9.2	9.3
WLAN	Wi-Fi 802.11 b/g, UPnP technology	Wi-Fi 802.11 b/g, UPnP technology	Wi-Fi 802.11 b/g	Wi-Fi 802.11 b/g,

Table 2. Compatibility Test Results

No	Protocol	Content Type	5800	N97	E71	E52
1	HTTP	3GP	O	O	O	O
2		MPEG4	O	O	O	O
3	RTSP	3GP	O	O	x	x
4		MPEG4	O	O	x	x

o = compatible , x = not compatible

3.3. Performance Experiments

Performance testing was conducted to observe the effect of several scenarios on system performance. System performance can be observed from execution time. This experiment is still using HTTP and RSTP protocols.

3.3.1 RTSP

The RTSP performance testing scenario is to measure the ratio of packets sent by the distance from the center of a WLAN access point. This scenario is used to observe the quality of video streaming with WiFi coverage from nearest to farthest.

Table 3. Scenario 1 RTSP Performance Testing

Video duration (Second)	Bit Rate (Kbps)	Bytes Sent (KB)	Packet Loss (%)
1	90	11	0
7	81	51	0
12	66	93	0
30	67	233	0
43	66	327	0
64	67	496	0
109	62	846	0
149	69	1.126	0
193	69	1.449	0

In this scenario, the performance testing is conducted using WiFi with the best signal (excellent or WiFi signal contains 3 bars or full) to the distance from the access point is ± 6 meters. This scenario used a large video file (Pocoyo 2MB with 3GP video content type). The result is shown in Table 3. As Seen at Table 3, the average bit rate is 70.7 Kbps. During the experimen, there is no packet loss during data transmission because the signal quality is excellent.

In the second scenario, the performance test was conducted using a Wi-Fi signal with 2 bars signal, and the distance from the access point is ± 20 meters. The result of Scenario 2 RTSP test performance is shown in Table 4. As Seen at Table 4, the average bit rate of the system is 68.167 Kbps. During the performance experiment in scenario 2, there is small packet loss during data transmission, this happens because the signal quality is moderate.

In the scenario 3, the performance tests are conducted using Wi-Fi with poor signal (Low or Wi-Fi signal contains 1 bar) with distance from the access point is ± 36 meters. This scenario uses the same file with scenario 1. RTSP test performance scenario 3 results are shown in Table 5. Table 5 shows the average bit rate is 82.08 Kbps. The performance experiment in scenario 3 shows that almost entirely contained packet is loss during data transmission, this happens because the signal quality is not good or bad.

Table 4. Scenario 2 Performance Testing RTSP

Video Duration (second)	Bit Rate (Kbps)	Bytes Sent (KB)	Packet Loss (%)
1	67	11	0
12	66	84	0
23	65	167	30
33	66	250	33
44	66	329	50
54	63	412	55
76	71	580	9
107	61	827	0
128	75	981	25
149	72	1.116	0
170	61	1.274	0
201	85	1.508	0

Table 5. Performance Test Scenario 3 RTSP

Video Duration (Second)	Bit Rate (Kbps)	Bytes Sent (KB)	Packet Loss (%)
6	113	37	59
16	104	117	0
27	95	199	91
38	99	328	45
59	68	446	0
71	61	544	0
91	102	714	42
102	61	870	14
144	65	1.048	69
164	68	1.163	19
186	63	1.394	15
227	86	1.709	33

3.3.2 HTTP

The HTTP protocol performance experiment compared the size of the video file with the data transfer time. Data transfer time is the time needed by the application to completely download files on request by the client to the content server.

Table 6 and Figure 12 show that the greater size of the video files requested the longer time required to download the file before running at the client side. The use of the HTTP pr otocol is not influenced by the bandwidth when streaming video delivery process is underway. It still works the same way as when using RSTP protocol.

Table 6. Performance Testing Using HTTP Protocol

No	File Size (Byte)	Data Transfer Time (Second)
1	1086681	2
2	2228424	3
3	8460023	6
4	9008493	7
5	9872333	8
6	11444337	10

7.3.3 RTSP vs. HTTP

This experiment compares the performance between RTSP and HTTP. This experiment will compare the size of video files, which will be streamed, to playing time of RTSP and HTTP protocols. Play time is the time needed by the request and response mechanism to get the video running. Table 7 shows the performance comparison test of RTSP and HTTP.

Table 7. Comparison of Performance Testing RTSP and HTTP

File Video No	Size file (Byte)	Time Play HTTP (second)	Time Play RTSP (second)
1	8460023	6	5
2	9008493	7	5
3	9872333	8	5
4	11444337	10	5
5	15776834	15	5
6	18843122	18	6

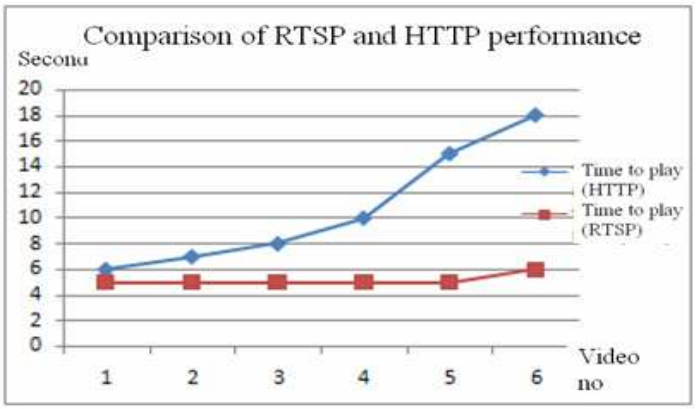


Figure 13. Comparisons of RTSP and HTTP

Based on the Figure 13, it can be seen that the use of HTTP protocol, the larger the size of video files then the longer time required performing Play action. This is because the HTTP protocol must first download the video file from content server completely. On the other hand, the RTSP protocol plays the file as soon as part of video file downloaded. Therefore, the process to deliver streaming video on the average has the same time for each size of different file video. This is because by the use of RTSP protocol, RTSP server manages all the data transmission process in the form of an input stream to the client. The process of data transmission in bits and pieces is what causes the process Play the RTSP protocol run faster than the HTTP protocol.

4. Conclusion

Based on the experiments conducted, it can be concluded that in the process of video streaming, RTSP is better if the streamed video files are very large, while HTTP is used if a

small sized files. This is because the RTSP works by running piece by piece of large file, therefore it does not take longer to play streaming video files. In addition, the use of RTSP protocol gives benefits for mobile phones, because it does not need large storage media to play large video files.

References

[1] Sangok K, Kanghee L, Zhefan J, Hyunchul B, Sangwook K. *Streaming Player Support Protocol Adaptation and Independent Operating System*. Fourth Annual ACIS International Conference on Computer and Information Science. Jeju Island, South Korea. 2005: 194-197.
[2] Min Q, Zimmermann R. An Adaptive Strategy for Mobile Ad Hoc Media Streaming. *IEEE Transactions on Multimedia*. 2010; 12(4): 317-329.
[3] Vazquez M, Vincent P. *A Mobile Audio Messages Streaming System*. Proceedings of the Euro American conference on Telematics and information systems (EATIS). Faro, Portugal. 2007: 1-4.
[4] Baumgart AS, Knapp H, Schader M, Mill S. *A Platform-Independent Adaptive Video Streaming Client for Mobile Devices*. The 7th IFIP International conference on Mobile and Wireless Communications Networks. Marrakech, Morocco. 2005: 11-15.
[5] Brandt J, Wolf L. *Adaptive Video Streaming for Mobile Clients*. Proceeding of the 18th International Workshop on Network and Operating Support for Digital Audio and Video (NOSSDAV). Braunschweigh, Germany. 2008: 113-114.
[6] Goyal V. Pro Java ME MMAPI Mobile Media API for Java Micro Edition. Second Edition. New York: Apress. 2006.
[7] English R, Schweik CM. *Identifying Success and Tragedy of FLOSS Commons: A Preliminary Classification of Sourceforge.net Projects*. First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS). Minneapolis, MN. 2007: 11-14.
[8] Shiow-yang W, Jungchu H, Chieh-Ming C. Headlight Prefetching and Dynamic Chaining for Cooperative Media Streaming in Mobile Environments. *IEEE Transactions on Mobile Computing*. 2009; 8(2): 173-187.
[9] Moxian L, Tsekleves E, Cosmas JP. *Semi-automatic creation of graphically-rich mobile Television services and applications using an XHTML browser and J2ME*. IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB). Shanghai. 2010: 1-7.