International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 11, November - 2013

Design and Implementation of Live Streaming using Embedded Linux

A.N.T.Sowjanya

(Aurora's Technological and Research Institute/ JNTUH, India)

N. Nirmala Devi

Associate Professor, ECE Department
(Aurora's Technological and Research Institute/ JNTUH, India)

Abstract:

In recent years there is need to improve the way the video content is delivered over the Internet. As multimedia consumption became significant part of the overall traffic usage, coupled with the emergence of mobile browsing content, providers needed a reliable and robust way to deliver content to users. Adaptive streaming is the new trend in video delivery on the Internet and is expected to be supported by consumer electronic devices such as CD players and DVRs. Standard solutions have been around for a couple of years and standardization efforts are in the final stage of implementation. To make this platform successful, optimized content preparation algorithms are needed. We propose a method to optimize the video content for better delivery. Resulting video stream is provides better experience for the users. At these large consumption rates, even modest reductions in video bitrates would result in significant reduction in infrastructure costs. Our system is used to design to capture a continuous stream of videos like live TV channels and videos are stored inside internet by using HTTP protocol.

Keywords*:streaming,HTTP protocol,segmentation,optimization.*

1. INTRODUCTION

In recent years there is need to improve the way the video content is delivered over the Internet. Video can be termed as the "next killer application" for the bandwidth it occupies. Multimedia content is the maximum proportion of the traffic being encountered in the day-to-day life.

There is need to provide a reliable and robust way to deliver content to users. The video stream has to adapt to the varying bandwidth capabilities in order to deliver the user a continuous Video stream without discontinuity at the best possible quality for the required moment.

At these large consumption rates, even small changes in video bitrates would result in significant reduction in infrastructure costs. It is essential to investigate possible optimizations that would introduce savings in bandwidth related to video content.

2. LITERATURE SURVEY

Most popular implementations available for streaming are Smooth Streaming, HTTP Dynamic Streaming [1], and HLS (HTTP Live Streaming) [2]. All solutions have the same underlying architecture - server side contains segmented video sequences and manifest file that describes content properties and lists segment URLs.

Segments are offered at different bitrate levels which allow switching between bitrates when needed. Client side decides what bit rate segment to download based on the current bandwidth conditions and other factors, such as CPU and memory usage.

At the highest level, Real-Time Entertainment (comprised mostly of streaming video) remains the largest traffic category on virtually every network we can examine, a continuous trend that we expect to observe into the foreseeable future.

The efforts for optimizations for adaptive streaming started with the first draft versions of the DASH specifications. The proposed models exclusively concentrate on the performance related to underlying network. All of these proposed solutions either require significant additions to common workflow or they require client side modifications.

Streaming is a technique for transferring data so that it can be processed as a steady and continuous stream. Streaming technologies are becoming increasingly important with the growth of the Internet because most users do not have fast enough access to download large multimedia files quickly. With streaming, the client browser or plug-in can start displaying the data before the entire file has been transmitted.
For streaming to work, the client side receiving the data must be able to collect the data and send it as a steady stream to the application that is processing the data and converting it to sound or pictures. This means that if the streaming client receives the data more quickly than required, it needs to save the excess data in a buffer. If the data doesn't come quickly enough, however, the presentation of the data will not be smooth. There are a number of competing streaming technologies emerging.

Our work addresses the problem of optimizing content encoding to reduce the bitrates used in video services. Our approach can be used with other delivery optimization solutions for HTTP streaming. It is also generally applicable to any adaptive streaming platform and doesn't depend on the specific coding or delivery method that might be implemented.

3. CONTENT PREPERATION

The main part of adaptive streaming workflow is content preparation, since it should be done in the most generalized fashion that is going to allow the best user experience in any network conditions. The performance of adaptive streaming can be increased by using good algorithms which are effective. Current implementations use approaches in which all content is encoded using simplified H.264 baseline profile.H.264 Scalable Video coding extension
(SVC) is used for content preparation instead of H.264 Advanced Video Coding (AVC) in order to allow better performance of underlying network.

3.1 OVERVIEW OF THE ALOGRITHM:

The algorithm used for the implementation is H.264 Scalable Video coding (SVC). SVC (Scalable Video Coding) [3] is an extension to the H.264 codec standard that is used by most of today's video conferencing devices. SVC video technology allows video conferencing devices to send and receive multi-layered video streams composed of a small base layer and optional additional layers that enhance resolution, frame rate and quality.

The term "scalability" refers to the removal of parts of the video bit stream in order to adapt it to the various needs or preferences of end users.
The objective of the SVC standardization is to enable the encoding of a high-quality video bit stream that contains one or more subset bit streams that can themselves be decoded with a complexity and reconstruction quality similar to that achieved using the existing AVC design with the same quantity of data as in the subset bit stream.
Profile of H.264 supports three basic slide coding types: 1) I-slice: intra-picture coding from neighboring regions 2) P-slice: intra picture coding and inter predictive coding with one prediction signal for each predictive region. 3) B-slice: intra picture and inter picture bipredictive coding with two prediction signals that are combined with a average to form region prediction.

4. HARDWARE:

The block diagram for the implementation can be shown as below:

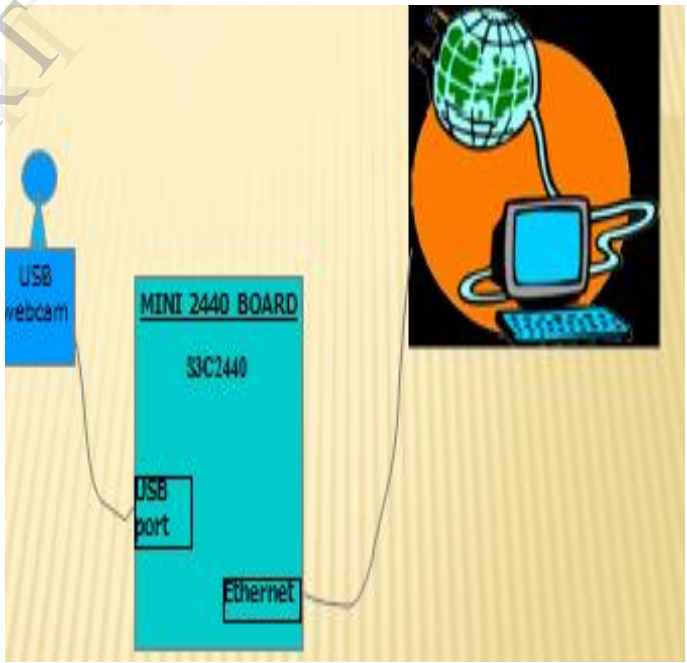

Figure: Block diagram of Implementation

The processor being used for implementation is ARM9 (S3C2440). SAMSUNG's S3C2440A is designed to provide hand-held devices and general applications with low-power, and high-performance microcontroller solution in small die size.
The raw data is recorded using the USB camera and streamed using the Ethernet. The coded data is stored

is stored in the buffer and the ARM board acts as the server board for the application.
Linux is the operating system being utilized and programmed in C language. Linux refers to the family of Unix-like computer operating systems using the Linux kernel.

Figure2: USB camera

Ethernet is a family of computer networking technologies for local area networks (LANs) commercially introduced in 1980. Standardized in IEEE 802.3, Ethernet has largely replaced competing wired LAN technologies.

Systems communicating over Ethernet divide a stream of data into individual packets called frames. Each frame contains source and destination addresses and error-checking data so that damaged data can be detected and re-transmitted.

Data rates were periodically increased from the original 10 megabits per second, to 100 gigabits per second.

The USB video device class (also USB video class or UVC) is a USB device class that describes devices capable of streaming video like webcams, digital camcorders, transcoders, analog video converters and still-image cameras.

The latest revision of the USB video class specification carries the version number 1.5 and was defined by the USB Implementers Forum in a set of documents describing both the basic protocol and the different payload formats. Video4Linux or V4L is a video capture application programming interface (API) for Linux, supporting many USB webcams, TV tuners, and other devices. Video4Linux is closely integrated with the Linux kernel.

Video4Linux was named after Video for Windows (which is sometimes abbreviated "V4W"), but is not technically related to it. V4L had been introduced late into the 2.1.X development cycle of the Linux kernel.

V4L1 support was dropped in kernel 2.6.38. V4L2 is the second version of V4L.

Video4Linux2 fixed some design bugs and started appearing in the 2.5.X kernels.

Video4Linux2 drivers include a compatibility mode for Video4Linux1 applications, though the support can be incomplete and it is recommended to use Video4Linux1 devices in V4L2 mode.

5. RESULTS:

The streaming shows the initial streaming before encoding

Figure3: Initial streaming

The streaming result after encoding we can observe increase in the speed of the streaming as shown below

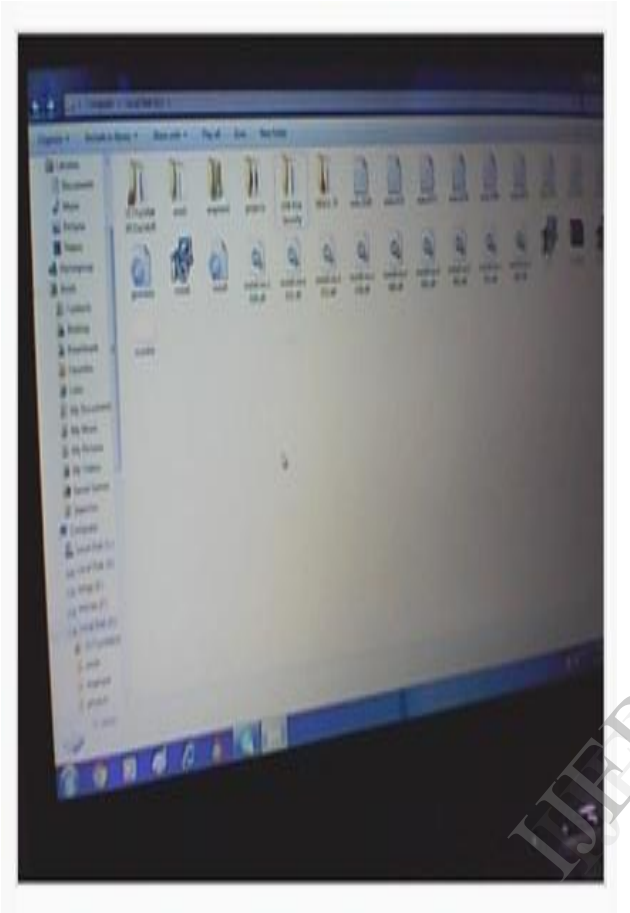

Figure4: Live streaming after encoding

6. FUTURE ENHANCEMENTS

1)The possible future enhancements are the protocol now being used is HTTP protocol and the video can be streamed only locally to make it stream globally and make it available everywhere it can be streamed using RTP or RSTP protocol.

2)Another future enhancement option can to increase the memory capacity of the application.

7. CONCLUSION

We presented a novel approach for encoding and segmentation of video content intended for adaptive streaming Over HTTP.

Optimized algorithms are essential for content preparation as adaptive streaming is gaining more popularity on the Internet. Coupled with other techniques for delivery and user side adaptation this approach can introduce significant savings.
The algorithm is also compatible with all standards since its implemented using H.264 encoder and segmentation is done by using longest segment duration attribute which is part of both DASH and HLS specifications.
Also, it is applicable in both on demand and live streaming scenarios. Since no changes are introduced to adaptive streaming standards, the algorithm can be used as-is.
In the case where simplified GOP structure is needed only a pre-processing step is needed in order to supply list of optimal I-frames positions to encoder.

8. REFERENCES:

[1] S. Akhshabi, A.C. Begen, and C.Dovrolis, "An experimental evaluation of rate adaptation algorithms in adaptive streaming over HTTP," *Proc. of the second annual ACM conference on Multimedia systems*, pp.157-168, Feb. 2011.

[2] R. Pantos, and W. May, "HTTP Live Streaming" *Draft v.07, IETF*, Sep. 2011.

[3] IEEE Transactions on circuits and systems for video technology, VOL. 17, NO. 9, SEPTEMBER 2007. SCHWARZ et al.: OVERVIEW OF THE SCALABLE VIDEO CODING EXTENSION.

[4] Sandvine Intelligent Broadband Networks, "Internet Phenomena report", *Whitepaper report*, Sep. 2011.

[5] Detlev Marpe, Thomas Wiegand, and Stephen Gordon.H.264/MPEG4-AVC _delity range extensions: tools, pro_les,performance, and application areas. In ICIP (1), pages 593{596, 2005.

[6] Heiko Schwarz, Detlev Marpe, and Thomas Wieg. Overview of the scalable H.264/MPEG4-AVC extension. In Proceedings of the IEEE International Conference on Image Processing, ICIP '06, pages 161{164, 2006.

9. AUTHORS:

1. A.N.T.SOWJANYA submitting in partial fulfillment of the requirements for the award of the degree of Master of Technology. Pursuing my M.TECH from Aurora's Technological and Research Institute.

2. N. Nirmala Devi (B.TECH, M.TECH) Associate Professor ECE department Aurora's Technological and Research Institute.