SMART SURVEILLANCE CAMERA USING AN EMBEDDED WEB SERVER

Apurv D. Nerlekar Saket P. Mandke Yash A. Kelkar
apurvnerlekar.comp2@mmcoe.edu.in spmandke@gmail.com yashkelkar.comp2@mmcoe.edu.in
Marathwada Mitra Mandal's College of Engineering-Pune 411052

***Abstract*— Security is a major concern nowadays. We have many gadgets to overcome this problem. One such gadget is a surveillance camera. We can keep an eye on a remotely located area by sitting at our work desks. A surveillance camera connected to a server streams live videos over the internet which can be watched from anywhere around the globe. A little study on such devices gave us an inspiration to develop an improved version of that same product. Our study revealed that servers used for such surveillance cameras were never used to their fullest capacity. So we had a scope to improve in terms of optimum hardware utilization and cost. Also streaming videos over the internet is quite a hefty procedure. We also used image streaming instead of video streaming to make the product work on connections with lower bandwidths. Such surveillance footages have to be stored for a number of days. Maintaining such archives requires a lot of storage space. We also devised a smart idea here to store only those images that are relevant to the main security issue. To implement all these solutions we have integrated technologies like virtualization, embedded systems, web technology and image processing into one product. The product will prove to be highly efficient in terms of utilization of hardware and cost.**

***Index Terms* —.Arch Linux, Embedded Systems, Image Processing, OpenCV, Streaming, Virtualization, Web Server**

I. INTRODUCTION

Surveillance cameras have become a common security tool at almost all public places. One of the high-tech features of these cameras is that you can view the live streaming over the internet. But that involves a lot of cost for the server and the internet connection. Also a lot of storage space is required to maintain an archive of such surveillance videos. The basic idea behind this project is to cut down on the set-up cost for such essential cameras and to maintain archives using lesser storage space. To achieve this goal we have integrated the features of virtualization, embedded technology, web technology and image processing into one single product. The beauty of this project is to improve upon an existing tool to make it more efficient and economical.

A study of the existing technology reveals that most of the servers are used only up to 5-10% of their capacity. Increase in load on the servers results in server failure. Hence optimum utilization of the hardware resources is not achieved. This statistic was one of the main motivations behind this project. Also streaming videos over the internet is quite an expensive task in terms of the required bandwidth and the process of streaming. Comparatively it is economical to stream images. Hence another point of motivation was to stream images in such a way that it would seem like a motion picture which would be good enough to serve the purpose of security. Maintaining an archive for surveillance cameras is one of the most important things. However storing videos costs a lot of storage space. Image processing can be used to store only the relevant footages and discard the other parts. We can save a lot of storage space and also maintain archives for a longer duration.

With all these points in mind we came up with this idea of a surveillance camera that uses an embedded web server to stream images rather than videos and that stores footages using smart image processing technologies.

II. CLIENT SERVER INTERACTION

Client Interfaces

- Login authentication page
- User home page
- User control options page
- Server configuration page

This system can consist of multiple users accessing the server concurrently. However, they are classified according to their permissions and rights as general users and administrator. The information about these users is stored in a simple and a light weight database, namely SQLite. SQLite is an ACID compliant embedded relational database management system contained in a small C programming library.
It has a dynamically and weakly typed SQL syntax
SQLite is a popular choice for local/client storage on web browsers.
The existing systems have their databases which has a larger memory requirement The library files of SQLite have a

memory requirement of ~275 kB hence making the database light weight with respect to memory.

To keep the system secure the username and passwords assigned to the users are stored in the database in an encrypted form. Encryption is done using the inbuilt php function md5. The username and passwords are not kept in the database in plaintext form so that even the administrator cannot view the usernames and passwords.

The interaction between the client and server takes place using the Hyper Text Transfer Protocol. The client connects to the server using an HTTP request. The server responds with corresponding HTTP response.

III. Embedded web server

A. Overview

A dedicated web server has its downsides. As already stated most of the hardware resources in the servers remain unutilized[4]. To overcome this problem embedded web server is a good solution. An embedded web server has very less storage space and thus it makes it imperative to transfer the images to the database. Also the embedded web server will tend to be more stable and thus problems of servers getting crashed will be reduced. An embedded device will require low power. The camera can be directly interfaced to the embedded board.

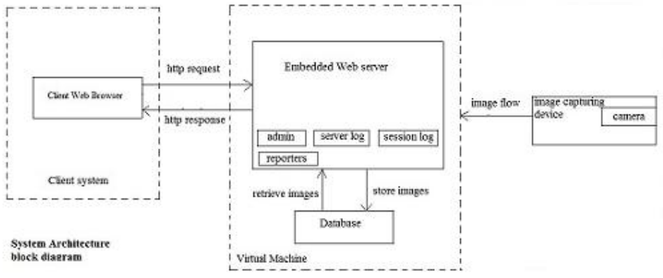
B. System Architecture:

The system Architecture is basically divided into the client block and the server system block.

The client block consists of the system from which the user connects from a remote location. The interaction is achieved using a web browser.

The other block consists of the embedded web server along with the Database and the image capturing device, the USB camera. This complete block is developed and built on a virtual machine with Linux platform.

Once the software is engineered, it can be ported on any of the embedded hardware such as ARM or x86 processors.



C. Implementation

Embedded system design requires special considerations due to its limited hardware resources and small footprint software requirements. In particular, the embedded imaging/video system design possesses even tougher limitations on the hardware/software resources while maintaining the acceptable performance. A conventional design and implementation very often lacks the capability to handle image/video related applications because of
(1) the large amount of video/image data to be stored demands large size memory unit in the multiple Megabyte range. This can be very challenging for low cost CPU[3], and (2) the speed to communicate with a video capture device.

Here, virtualization was a successful solution to the above problems[2]. The concept of virtualization provides an opportunity to the software development team to design and implement software product simultaneously before the actual compatible hardware is been chosen. Virtual machines can be instantiated multiple times. Thus, it is possible to have multiple independent logical subsystems within a single physical machine. Programs which exist in different subsystems cannot interfere with each other, except through interfaces which are provided (and controlled) by the virtualization environment.

It seems logical to apply virtualization to embedded systems. However, this raises the question whether it can simply be reused" as it is", or if any conceptual changes must be made to make it applicable to embedded systems.

Arch ISO-

Archiso is a small set of bash scripts that is capable of building fully functional Arch Linux based live CD/DVD and USB images. It is a very generic tool, so it could potentially be used to generate anything from rescue systems, to install disks, to special interest live CD/DVD/USB systems, and who knows what else. The heart and soul of Archiso is mkarchiso. All of its options are documented in its usage output.. Due to recent changes, Archiso will now automatically create ISO images that are also USB images. Separate CD/USB targets are therefore no longer necessary. We can use this archiso to mount the software on our hardware. Once we have created an image of the arch system on which we have done all the coding, we can take it on an USB device. This USB device can be interfaced with the hardware board. The USB can be made bootable. Now once the hardware is powered on the arch image will be booted and the hardware will now run the arch system. Hence the name 'embedded web server'.

Cross compilation

The working platform as being virtual machine the final product needs to be deployed on some embedded device. To make our code compatible with the embedded device, cross compilation will be required for us to make the code feasible with the device. Cross compiler is a compiler capable of creating executable code for a platform other than the one on which the compiler is run. Cross compiler tools are used to generate executables for embedded system or multiple platforms. It is used to compile for a platform upon which it is not feasible to do the compiling, like microcontrollers that don't support an operating system. It has become more common to use this tool for Para-virtualization where a system may have one or more platforms in use.

IV. Streaming Surveillance Videos

Computer video files are basically a number of still images called frames combined sequentially into one file. When the file is played on a player such as Windows Media Player, it goes through the video file and displays each frame sequentially in quick succession to create the illusion of motion similar to a video film rolling through a movie projector [1].

When a file is transferred, frames are continuously delivered from the computer transferring the video to the computer playing it. Each frame is displayed as it is received.

For example, consider a computer connected to the Internet using a dialup modem. If the modem is connected at 40 kbps, it could receive 5,000 bytes (5 KB) of data per second. If each frame of the video was only 5K then the modem could only receive 1 frame per second.

Commercial motion pictures are 24 fps (frames per second); television is 30 frames per second. The more frames per second (fps), the smother the video playback appears to the viewer. So, a 1 fps video is a very slow and choppy video. The number of frames per second (fps) is also called the frame rate.

With a higher bandwidth connection, more frames per second could be received. With a 128 kbps ISDN connection for example, 32 5K frames could be delivered per second.

But, a 5K image or frame is not very big. A small 320x200, 16 bit JPEG file can easily be 20K in size. So, for the modem connected with only 40 kbps of bandwidth, it would take 4 seconds to receive only one frame of the video! At that rate, the video would degrade into a slide show, and not be a video at all.

Thus to enable the users to use this product on an internet with low bandwidth we are providing users with the live surveillance footages at the rate of 1 frame for every two seconds. The video might not match the commercial video standards but still is good enough to maintain security.

To capture the live images from the camera, very developed and advanced set of library functions are implemented. The OpenCV Library is a collection of low-overhead, high-performance operations performed on images.

Why OpenCV---?

The OpenCV Library is a way of establishing an open source vision community that will make better use of up-to-date opportunities to apply computer vision in the growing PC environment. The software provides a set of image processing functions, as well as image and pattern analysis functions. The OpenCV Library has platform-independent interface and supplied with whole C sources. OpenCV is open. The various functions of opencv help in querying a frame, opening an image in a window, save an image and load an image. Some of the functions are:

```
int cvNamedWindow(const char* name, int flags=CV_WINDOW_AUTOSIZE);
```

The function cvNamedWindow creates a window which can be used as a placeholder for images and track bars. Created windows are referred by their names.

```
void cvShowImage( const char* name, const CvArr* image );
```

The function cvShowImage shows the image in the specified window. If the window was created with CV_WINDOW_AUTOSIZE flag then the image is shown with its original size, otherwise the image is scaled to fit the window.

```
int cvSaveImage( const char* filename, const CvArr* image );
```

The function cvSaveImage saves the image to the specified file. The image format is chosen depending on the filename extension,

V. Synchronization

Synchronization will play an important role in this product. Since it is an embedded web server it will have minimum storage capacity. Hence once it has captured an image it will immediately look to stream it online or store it on the database. Both these actions will be taking place very quickly and almost at the same times. Hence it will be necessary to avoid concurrency among the two processes. Consider that the captured image is being streamed online. The client browser will be copying this image from the server. If at the same time the server tries to store a captured image on the same image file, it will result in a corrupt image getting displayed on the clients' browser. To avoid this we have used system locks. Each process can be given a shared lock or an exclusive lock. This can be done by using techniques as below.

```
int flock(int fd, int operation);
```

flock()-Apply or remove an advisory lock on the open file specified by fd. The parameter operation is one of

the following:

LOCK_EX:

Place an exclusive lock. Only one process may hold an exclusive lock for a given file at a given time.

LOCK_UN:

Remove an existing lock held by this process.

Hence we can give the server an exclusive lock so that while it is saving a new image no other process can access it. On the other hand we can give the client side application a shared lock. So that it can read the image while no writing process is taking place on that particular file. Copying the image file to the database can also be given a shared lock. Browser application and the process of copying image to database both these process can run in concurrency.

VI. Smart Image Processing

It is sometimes of interest to process a single sub region of an image, leaving other regions unchanged. This is commonly referred to as region-of-interest (ROI) processing. Image sub-regions may be conveniently specified by using Mathematical Graphics primitives, such as Point, Line, Circle, Polygon, or simply as a list of vertex positions.

As image processing can take a long time, we can use a "region of interest" (ROI) to quickly preview the results of an operation. Most operations are designed so that they can operate on a ROI in the original image. This is useful if you want to apply a time-consuming operation to a large image--you can first test it on a small ROI, and then when the parameters are correct, apply it to the whole image.

Once the browser receives the captured images from the server, a certain region of the image frame can be selected and its co-ordinates can be passed back to the server for image cropping. the confidentiality of the image data remains intact as processing is done at the server end. JavaScript code is generally used to create a selection over an image and send the x & y coordinates of that selection to another script. One way is to use PHP to process the image and do the actual cropping, but any method can be used to do it.

The image is displayed, and the user clicks first in the upper-left of the region they want to crop to, then in the lower-right. When the user submits the selected area, the browser is with GET values attached like so:

```
<yoururl>?xcoord1=<x1>&ycoord1=<y1>&width=<width>&
height=<height>&image=<image>
```

Where "<your url>" is what you supplied with the Crop Image button, <x1> and <y1> are the x, y coordinates of the top-left corner of the selection and <width> and <height> are the pixel dimensions of the selection area, and <image> is the path to the image (URI).

The parameters from the browser are saved into variables in the server and the image is cropped accordingly. Since HTTP is a stateless protocol, it is necessary to save the co-ordinates of the area with the client side script so that it can be referred in near future.

The image frame can be than cropped by using standard GD Library functions with Php APIs. These functions help in reading the fimage file and crop the image by providing the required height and width of selected region and the x-y coordinates of the actual image.

```
$srcImage=CreateImageFromJpeg(“filename”);
```

-Create a new image from file or URL

```
$destImage=CreateImageTrueColor($width,$height);
```

- Create a new true color image

```
ImageCopy($srcImage,$destImage,$new_x,$new_y,$x,$y,$wi
dth,$height);
```

-Copy part of an image.

VII. Summary

We have used varied technologies like virtualization, embedded systems, web technology and image processing. The basic idea was to reap the positive aspects of all these technologies and come up with a product that would be overcome the shortcomings of the existing product. We studied the performance of the existing systems, found out the areas where improvement is possible. We then thought of ways to overcome these problems. After implementing the above given solutions we could see that the product has overcome all its drawbacks. Thus we have come up with an old product but using new technologies to deliver optimum performance.

Image processing techniques can also be used to cut down on the storage space requirement. This can be done by comparing images. Each frame is compared with its preceding frame. If any difference is found in those two images, then the latter is stored in the database. This process can be carried out for every captured frame. This ensures that archives are maintained only when some activity takes place in our region of surveillance. By this we can maintain archives for longer periods of time.

Also, a further research is possible on the basis of this system. The captured data can be processed using different processing techniques before the user actually handles it. An in-depth research about the various technologies implemented can be done using this portable system.

ACKNOWLEDGEMENTS

We are highly indebted to our internal guide, Prof. Mrs. Meenal Mungi, Computer Department, MMCOE, Pune, for her guidance and constant supervision as well as for providing necessary information regarding the project & also for her support in life time of the project. Our thanks and appreciations also go to our college, Marathawada Mitra Mandal’s College of Engineering and people who have willingly helped us out with their abilities

We would like to express our gratitude towards our parents & member of Umitas Technologies for their kind co-operation and encouragement which helped us in this project.

REFERENCES

IEEE Papers:

[1] Chen Xi, Wang Wei-li, Ma Jiang-hua, ”The design of embedded image capturing system”, 2009 Second International Conference on Intelligent Networks and Intelligent Systems.

[2] Gernot Heiser, ”The Role of Virtualization in Embedded Systems”, First Workshop on Isolation and Integration in Embedded Systems (IIES’08) April 1, 2008, Glasgow, UK.

[3] Hiromasa Shimada, Alexandre Courbot, Yuki Kinebuchi, Tatsuo Nakajima,"A Lightweight Monitoring Service for Multi-Core Embedded Systems", 2010 13th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing

[4] Liting Cao, Jingwen Tian and Dahang Zhang,"Networked Remote Meter-Reading System Based on Wireless Communication Technology", 2006 IEEE International Conference on Information Acquisition August 20 - 23, 2006, Weihai, Shandong, China.

[5] Ma Ke, Song Changxin , "Embedded Network Management System Design Based on WEB", 2009 International Conference on Signal Processing Systems.

[6] Mo Guan, Wei Wei, Ying Bao , "A Monitoring System Based on Embedded Internet Technology for Embedded Devices", 2008 International Conference on Computer Science and Software Engineering.