AN EFFICIENT SOFTWARE ARCHITECTURE FOR VIDEO MANAGEMENT SYSTEM

Mitesh Patel

M. Tech in Embedded and VLSI Systems.
U. V. Patel College of Engineering, Kherva, Gujarat, India.

ABSTRACT

Video Surveillance System is rapidly growing domain as organizations demands to safeguard physical and capital assets. As the database increases and number of features increases in video management systems the whole system’s efficiency decreases. Video management systems contains many modules like live, playback, carousel etc. This paper addresses software architecture specially designed for video management system.

I. INTRODUCTION

Algorithms and data structures constitute the major design problems when sizes of video surveillance software increases. When Video Management systems constitutes many components the organization of the overall system leads into major design problems. The overall design of video management software can be presented in number of ways including descriptive terms, informal diagrams, modules interconnection languages, framework and templates for systems that serve the demands of particular domains and models of component.

II. DESIGN FLOW

Design flow of Video Surveillance System software encompasses various stages. From GUI to database, data travels to and from database to GUI. Data mainly travels in two forms, in structures and JSON form.

From GUI data is filled in Structures and is passed to controller section to call for specific API. Controller calls common communication framework where structure is serialized into light-weight data form JSON.

After that RESTAPI is called with requested JSON and particular API command and URL. RESTAPI with the help of CURL library calls CGI and gives JSON as a command line argument. Rest CGI calls Client to connect with Server using TCP socket. As server receives data and API command it transfers to Service that is running continuously and listens to service if any data is coming then it gives relative response and acts on data.
III. QT BASED GRAPHICAL USER INTERFACE

Qt is framework designed in C++. Framework includes extensive C++ libraries which helps engineers to develop and build highly performing application and GUI for several desktop and mobile OS. GUI can be made with any of the framework available in the markets. But the reason we go with the Qt is, it is faster and supports many embedded devices. Qt is framework written in C++. Qt supports Model/View architecture to handle data flow. Models represents classes that stores data. Data can also be retrieved from model as and when required. Views are readymade classes in Qt similar to widgets that helps to view data to user. Types of models and views can be varies depending upon how one wants to represent data. It may be list-view, table-view, or can be a simple combo-box. Example for Model of List-view is set to object of list view class. As and when data is changed in Model view shows those changes to user in display. Model is connected to external world of Qt to Controller’s API which returns data. Models also passes data as an arguments to API to controller.

IV. CONTROLLER

Controller connects GUI to external world. As and when GUI needs to get data, update, get list or delete some data, it calls respective controller functions which in turn calls Send and Receive function which sends data in structure format and receives data in structure format. Controller returns status of whether API is successful or not. With structured format data, controller also sends API command for specific API.

V. COMMUNICATION FRAMEWORK

Communication framework constitutes main block of whole video management software architecture system. Communication Framework contains definition of Send and Receive function. Send and Receive function serializes data. The structured format data sent by GUI is converted into JSON (Java Script Object Notation). JSON is light weight compared to structure format. Hence it can be easily passed to RESTAPI and over network. URL (Universal Resource Allocator) is already defined in

Figure 1. Architecture
communication framework for each API. So, the command sent by GUI is used to get URL of specific API. The RESTAPI sends JSON requests data and URL to CGI and receives response in JSON format only. JSON response will get de-serialized into structure format and is returned to controller via send and receive function of communication framework which was called by controller.

VI. RESTAPI

REST stands for Representational State Transfer. It relies on stateless, client-server, cacheable communications protocol and in virtually all the cases, the HTTP protocol is used. RESTAPI function is called in send and receive function in communication framework. It passed JSON format request data and URL of specific API to CGI (Common Gateway Interface) application. RESTAPI uses Curl library to call CGI application which is already built on machine with IP address provided in URL.

VII. CGI (COMMON GATEWAY INTERFACE)

CGI is the part of the networked system with servers and client. CGI can communicate with other applications running on the server. Web Server calls existing or a running service or program with the help of a CGI. CGI application receives command as an argument in main function. A Socket is one end point of a two-way communication between two application running on network. Application contains TCP Socket which makes connection with TCP Server. At the other side TCP Server keeps listening to TCP socket, if there is any request from TCP socket to connect. If request comes then TCP server accepts connection and then TCP socket starts to send data. TCP server receives and sends data to Service which is running on same port.

VIII. SERVICE

Service is the significant block where whole software architecture which has some logic. Service mainly provides requirement made by GUI. Service creates connection with the PostgreSQL database. Service create PostgreSQL queries as per requirement of GUI and executes queries to database and make changes to it. Mainly four queries are used to alter tables. “UPDATE” query updates data when GUI edits some data in it.

IX. RESULT

The whole architecture provides efficient calling of various APIs. Large data can be saved and retrieved to and from database. Callback APIs takes much less time. Figure 2 shows a service running continuously which provides services related to particular module. As shown in figure 2 service is listing some list of maps from the database.

![Figure 2. Service Running.](image)

X. CONCLUSION

Video management system can use such architecture to rely on performance, error-free environment. The architecture designed is such that it can handle large data movements in single callbacks. The time-duration that the whole system takes is comparatively much less.
REFERENCES


[2]. Advances in Software Engineering And Knowledge Engineering. Vincenzo Ambriola, Genoveffa Tortora. ISBN 981-02-1594-0


