IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 16, Issue 5, Ver. I (Sep – Oct. 2014), PP 61-66
www.iosrjournals.org

Simulation of ONVIF Network Devices Using JAVA

Yogiraj Awati
(Bachelor in Computer Engineering/ University of Pune, India)

Abstract : *In today's world, cross platform integration of application plays important role to achieve economic viability and usability of the product. This paper describes technical details of the successfully implemented system whose roots are based on ONVIF platform. System enumerates simulation of various ONVIF capable video surveillance devices and delineates the communication that takes place between user and network devices. ONVIF network discovery and accessibility of various services has been wisely implemented which gives us lucid understanding regarding ONVIF specification*
Keywords: *Database, JAX-WS, ONVIF, SOAP, Web Service XML*

I. Introduction

This research work is in continuation with paper [1] published by the same author. The main incitement of ONVIF [2] specification is to achieve interoperability between various networking devices. Large organizations are equipped with multiple security cameras from various vendors. So it's a matter of concern to handle such huge number of vendor specific devices, to accesses their features and efficiency of the communication is definitely impeded without any standard protocol. Thus, ONVIF plays vital role in establishing common platform for all these network devices and manages device discovery and service accessibility. This paper focuses on simulating ONVIF based device system and achieving discovery and accessing services by remote client using JAVA platform.

II. Discovery Concept

Discovery of ONVIF compliant devices is based on multicast messages sent across the network. According to figure 1, whenever a device is connected to network, it broadcasts a multicast Hello message [3]. So receivers are able to percept that a device has been added to network. End users are able to view services of the devices by sending a unicast probe message to device which consists of type of the device like NVT, NVR etc and scope of the device like location, hardware of the device. On receiving probe message on device side, probe matching takes place and respective device details are sent back in probe match message. Whenever device wants to get disconnected, it simply sends Bye which apprises devices regarding the disconnection of the device from the network. This communication is based on SOAP protocol which is encoded in XML [4].

To adhere with ONVIF, discovery system is implemented in JAX-WS language which is a subset of JAVA that simplifies implementation of SOAP and concerned implicit conversions as required by the specification [7]. The services provided by the device are exerted in the form of web services. The communication between applications takes place through socket and port number [5]. Database has been used to store various configuration details about the simulated devices. Login capability is provided for user, in order to maintain security and integrity of the system.

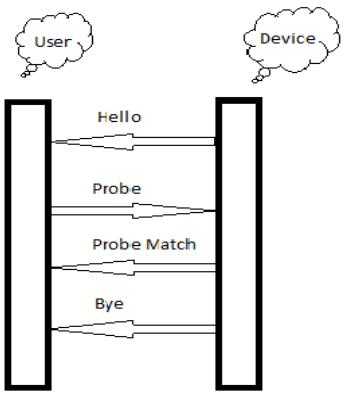

Fig. 1. Device Discovery

III. System Architecture

Figure 2 depicts conceptual structure of the application. Architecture is classified into four components:-

1) Database 2) Cloud 3) Client 4) Server

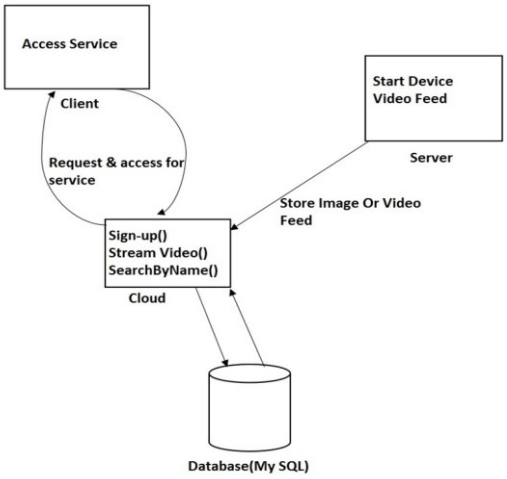

Fig. 2. System Architecture

3.1 Database

Any database can be considered depending upon your choice. Three tables are constructed. *Devicedetail* table consist all the configuration of the device like its id, type, building location. *Userdata* table is used for videostreaming. Runtime image at each millisecond is captured from camera and stored in the corresponding device id in the *Userdata* table. So this table is constantly updated by current image being fetched by the device. Current image replaces old image and hence it optimizes the size of the table. *Client* table is the third table in a row, used to store user authentication credentials like username, password.

3.2 Cloud

Cloud is the vital component of the system. It defines a webservice in which all the operations are defined. Whenever any client or server refers to a function, it just invokes that function by making a remote call. Various functions defined are login, searchByName, signup, streamvideo etc. Moreover it authenticates itself with the database and retrieves information for videostreaming which is stored by the device.

3.3 Client

Client is provided with simple graphical user interface to access the system. It enforces thin client concept. All the business rules are defined on cloud component. So the main function of client is just to take input from user and reply with corresponding output. Since we deploy Software Oriented Architecture, client accesses functionalities of the device from cloud application in the form of webservices.

The user interface provided takes input from user. An operation of the webservice which is running on cloud is invoked by client. Parameters are passed to this function. The implementation of this function is available on cloud and input passed, is processed on cloud and result is given back to client. Thus this logic enhances computation and optimizes performance on end user side.

To exemplify the process, when user will login to application, he will enter the necessary credentials. A login() function is invoked by client . So the control now passes to cloud component where definition of login() function is available. After execution of the function, if credentials are accurate, "Sign up successful" message is returned to client and then he can proceed else user has to renter correct credentials. Thus, it helps us to achieve lightweight client application.

List of available devices is populated by continuously scanning *Devicedetail* table with device status equals to 1. Whenever device leaves the network, implicitly its status value is changed to 0, and it is dropped from available device list.

3.4 Server

Server component of the software is provided for technical administrator. Server simulates multiple devices which are assumed to be ONVIF compatible. This application provides three main services: - Add device, Manage device and Start device. This component is completely hidden from the user and he is unaware

about its existence. Server provides flexibility to administrator by adding a device or updating configuration of device or else starting a new device in the network. Figure 3 depicts the user interface on server side.

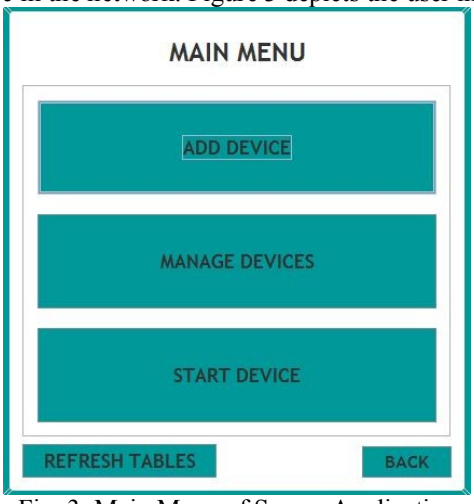

Fig. 3. Main Menu of Server Application

3.4.1 Add Device

If we want to add a device in the *devidedetail* table, we use this functionality. Figure 4 shows a device with NVT capability with its location specification is added to database. The types of device that can be assigned are NVT, NVR and Camera. For implementation purpose scope constitutes building name.

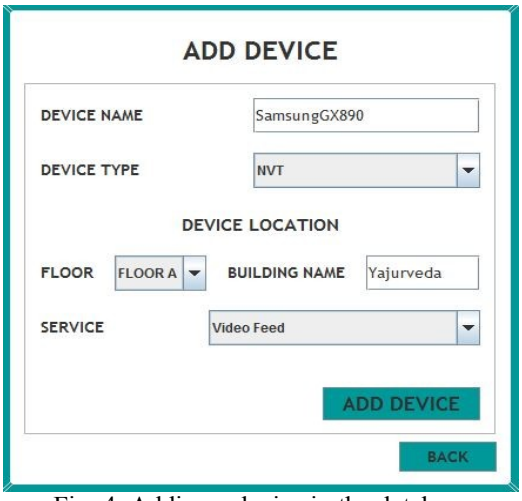

Fig. 4. Adding a device in the database

3.4.2 Manage Device

This functionality retrieves all the devices present in the database and one can select a device and update its capabilities. Concerned values of the attributes are reflected in the *Devicedetail* table.

3.4.3 Start Device

This functionality helps us to start a device and connect to network from the available list in the database. As shown in figure 5, administrator can select the resolution of video streaming or camera and concerned service is provided.

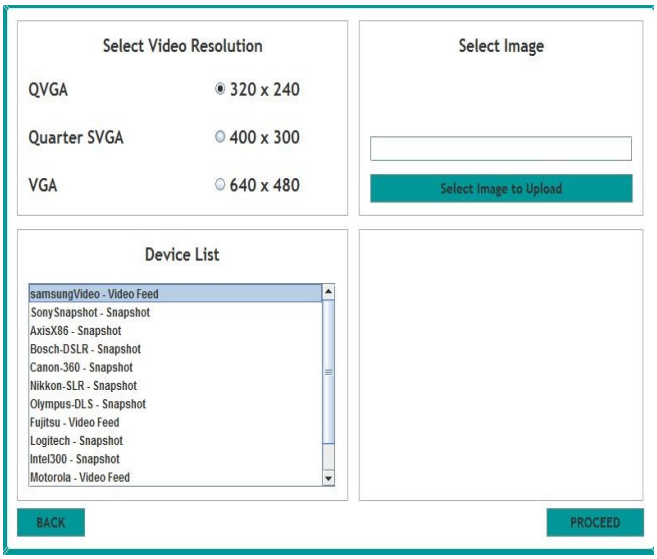

Fig. 5. To start a device

IV. Software Prerequisite

(1) NetBeans 7.1 (2) MySQL 5.1 (32-bit) (3) jdk6U_16 (32-bit) (4) JMyron Library (5) Establish LAN (6) Language used: - JAVA

V. Technical Description

5.1 JMyron

JMyron library used to access laptop's camera in JAVA programming language [6]. An object of JMyron is created to perform various operations on camera. start() and stop() methods are used to access camera. update() functionality captures next image from camera. Image that is available from laptop camera is in the form of RGB pixels.

5.2 VideoStreaming ()

This function is executed on NVT, NVR device which has video capability. Multithreading is used to capture video from camera and store in the database. The image that is available from camera is in the pixel format. But the pre-requisite of image storage is blob. So pixel to bytes transformation is needed. On successful conversion image is stored in the byte form in the database under *Userdata* table.

5.2.1 Algorithm

1. Initialize camera using JMyron object using a thread
2. Use timer function available in JAVA
3. For each iteration take the image from camera
4. Convert image from Pixel format to Byte format
5. Overwrite that image in *Userdata* table, corresponding to device id

5.3 FetchVideo ()

This function is executed on the user side. When user wants to stream video from a device that is available on the network this function fetches image data from *Userdata* table. The data is available in byte form. So byte to pixel conversion takes place and it is rendered on the screen. This procedure may be considered as exactly opposite to that of storing image in database.

VI. Implementation Algorithms

6.1 Server Side

1) On opening application, administrator may add, manage or start device.
2) Select resolution for streaming video or for snapshot capability of the device.
3) On proceeding further, press connect button which concludes that Hello message is broadcasted in the network. On background, the status field in *Devicedetail* table is set to value 1.
4) On starting device, VideoStreaming () function is invoked which continuously captures and stores image.
5) If the device is to be disconnected, a Bye message is broadcasted over the network and device leaves the environment. This can be technically achieved by setting status field of *Devicedetail* table to value 0.

6.2 Client Side

1) Client is logged in to the application by URL of the cloud application and credentials username and password. So whenever a function is invoked, URL is used to reach to definition of the function.
2) On successful login, available device list in the network is populated, by scanning *Devicedetail* table and fetching information of the devices whose status is equal to 1.
3) Client can select one of the available device and access its functionality.
4) FetchVideo () function is invoked for video streaming.
5) Client can also search devices by name which is an added feature of the project.
6) Searching functionality is enhanced by providing search by types and scope of the device.
7) At any point of time while accessing the service of the device, if the device crashes, an alert message is sent to client regarding disconnection of the concern device from the networks.

6.3. Cloud Side

1) Start cloud application in NetBeans environment. During initialization, authorised connection is established with the database.
2) Whenever a function is invoked on client side, parameters are extracted and respective function is executed.
3) On successful execution, result of the function is passed back to client.

VII. Result

Whenever client selects a device from available set of devices in the, he must be able to access its services. For instance figure 6 shows available devices. Nikkon-SLR is of video capability and user selects it, accordingly, user should be able to get specified service which is video streaming in this case.

Figure 7 shows the desired output. Whenever start feed is pressed, video streaming starts, if we press stop feed, video stops. In between if the device providing service crashes, an alert message is displayed to user and service stops. Thus adhering to ONVIF concept we successfully achieve output.

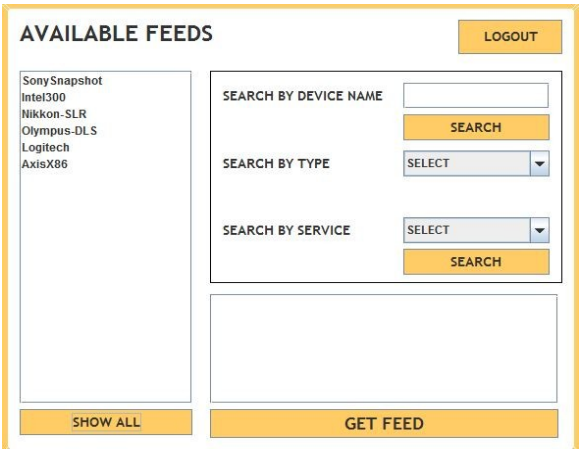

Fig. 6. Client accessing a device

Fig. 7. Client successfully receiving device service

VIII. Advantages

The system adumbrated is implementable in many practical situations. This project can be developed on large scale for video surveillance and will help to integrate vendor specific devices in an organization through presented single software. Motion detecting devices can adhere to ONVIF specification and also can be used in organizational security surveillance. It helps to achieve interoperability between IP-based physical security products regardless of manufacturer. System provides common base for a fully interoperable network implementation comprised of products from different network vendors.

IX. Implementation Considerations

System works well in LAN. For devices in different networks concerned changes in the project must be made compliant with the device proxy concept. IPv4 address is considered while implementation. Explicit changes are needed to make it compatible with IPv6. Since we are using JMyron library to access camera, while simulating various devices on same computer, only single device can take service of camera because single JMyron object blocks the cannel to camera and multiple instances are rejected to use camera simultaneously.

X. Conclusion

This paper elucidates the technical implementation of ONVIF network device discovery. ONVIF is a new platform in security and this application creates a paragon to understand how the working of ONVIF specification takes place. By simulating various devices, we are able to achieve exact environment so that further research could be made considering this system as the base platform. It explicates basics of ONVIF discovery and service accessibility to naive users. Moreover shifting functional processing from client application to cloud application definitely optimizes system performance.

XI. Abbreviations

ONVIF - Open Network Video Interface Forum
SOAP - Simple Object Access Protocol
JAX-WS – JAVA API for XML web services
XML – Extensible Markup Language
NVR – Network Video Recorder
NVT – Network Video Transmitter

References

[1] Yogiraj Awati, Rahul Gutal, Abhijeet Bhintade, Saurabh Taware, “Onvifsense: ONVIF Network DeviceAccessibility Application”, ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 3, Issue 4, April 2014
[2] ONVIF, ONVIF™ Core specification, Version 2.2. May, 2012. [Online] Available: http://www.onvif.org/specs/DocMap.html
[3] J. Beatty et al., XMLSOAP, Web Services Dynamic Discovery (WSDiscovery),April 2005 . ([Online]Available:http://specs.xmlsoap.org/ws/2005/04/discovery/wsdiscovery.pdf)
[4] T. Berners-Lee, et al, "Uniform Resource Identifiers (URI): Generic Syntax,"August 1998. ([Online] Available: http://www.ietf.org/rfc/rfc2396.txt.).
[5] "Port Numbers," February 2005. ([Online] Available: http://www.iana.org/assignments/port-numbers.)
[6] http://webcamxtra.sourceforge.net/reference.shtml
[7] http://docs.oracle.com/javaee/5/tutorial/doc/bnayl.html