CONTROLLING CAMERA AND PDU

O.J. Mokone, T. Gatsi, SKA South Africa NRF, South Africa

Abstract

The 64-dish MeerKAT radio telescope, currently under construction in South Africa, will become the largest and most sensitive radio telescope in the Southern Hemisphere until integrated with the Square Kilometre Array South Africa (SKA SA). This paper will present the software solutions that the MeerKAT Control and Monitoring (CAM) team implemented to achieve control (pan, tilt, zoom and focus) of the on-site video camera using the Pelco D protocol. Furthermore this paper will present how the outlets of the Power Distribution Unit (PDU) are switched on and off using Simple Network Management Protocol (SNMP) to facilitate emergency shutdown of equipment.

INTRODUCTION

The current running KAT-7 radio telescope (a MeerKAT prototype), which is running in the Karoo, is utilising three fixed cameras. Fixed cameras sometimes do offer appreciable performance in terms of resolution and image quality but do have amongst others a host of shortcomings for all their intents and purposes.

Despite them being more durable and needing less of mechanical service, the fixed cameras have the disadvantage of not covering a large area in the vicinity of the dishes and hence increases the cost of purchasing more fixed cameras in order to monitor the telescopes and the surrounding area.

To avert this challenge, the SKA SA MeerKAT team procured a Pan, Tilt and Zoom (PTZ) video camera. This Camera has been incorporated into the MeerKAT system because it can closely monitor the area under surveillance through the exploitation of the zoom-pan-tilt functionalities that are built within the camera.

The PTZ camera implements a state of the art technology, which is more appropriate for the MeerKAT project as it surveils a wide area around the dishes and allows for the pan, tilt and zoom capabilities. The PTZ camera is connected to the Power Distribution Unit (PDU). The PDU makes it possible to switch on/off the camera electronics that are connected to the specific PDU outlets remotely. The software that control and monitor camera was developed using Python programming language running on Ubuntu 14.04 LTS.

CAMERA SOFTWARE IMPLEMENTATION

Karoo Array Telescope Control Protocol (KATCP) [1] is a simple ASCII communication protocol layered on top of TCP/IP and is used for the monitoring and control of hardware devices. One of the most prevailing features of the KATCP protocol is its capability to interrogate KATCP servers for sensors. Interrogation of sensors [2] and requests, down to device level through KATCP, supports fluid run-time detection of system configuration, e.g. when a new sensor is added to any level of the system (including a hardware device), the rest of the CAM automatically discovers the change on-line on, and includes it in the monitor store and in its interfaces. Based on the functionality of the KATCP protocol, the video camera software had to implement similar technologies that respond to KATCP command messages.

Pelco D is a popular PTZ camera control protocol used in the CCTV industry. The camera control software uses Pelco D protocol over RS485, a TCP/IP physical interface. This is a client-server architecture, where the server is KATCP device server that connects directly to the camera device and the client is the CAM (Control And Monitoring) system software. One of the major roles of the server is to serve as a protocol translator; it translates Pelco D command messages to KATCP command messages and vice versa. Pelco D consists of 7 hexadecimal bytes. Table 1 (below) shows Pelco D message formats and their descriptions.

Table 1: Pelco D Message Format

Byte Number	Description
Byte 1	Synch Byte
Byte 2	Address
Byte 3	Command 1
Byte 4	Command 2
Byte 5	Data 1
Byte 6	Data 2
Byte 7	Checksum

Byte 1 is the synchronization byte, which is always fixed to FF. Byte 2 is logical address of the camera being controlled. Byte 7 is checksum, which is the 8 bit (modulo 256) sum of the payload bytes byte 2 through byte 6 in the message. The rest of the byte numbers depends on the commands that one is passing to the server.

PRESETS

The PTZ Camera is capable of storing and accessing up to 127 presets. These stored presets reside in a non-volatile memory within the camera. They can be accessed using an identity number from 1 through 127. The identity that is used to store a preset can subsequently be used to “go-to” to this preset, thus causing the camera to move to the previously stored configuration of pan, tilt,

ISBN 978-3-95450-148-9

zoom. Also, the previously stored presets positions can be cleared, which removes it from persistent memory.

To set the preset the CAM system (client) issues a KATCP message as "? preset-set 3" to the KATCP device translator. This KATCP message is then translated to Pelco D command as [FF, 0x01, 0x00, 0x03, 0x00, 0x3, 0x07]. The KATCP device translator relays the Pelco D message to the camera. The camera will then store the position that is currently in its persistent memory.

Query Position

The client has capability to query the position (PTZ) that the Camera is currently set to. To query zoom position the client will send a KATCP request message (? zoom-position) to the server and the server will then translate the KATCP message to Pelco D command message (FF, 0x01, 0x00, 0x55, 0x00, 0x00, 0x56), which will then send the Pelco D message to camera. The camera replies with the following message 0xFF, 0x01, 0x00, 0x5D, MSB, LSB, checksum. The translator then uses MSB (most significant bit) and LSB (least significant bit) to convert to the zoom position that the lense is currently at and send it to the client.

POWER DISTRIBUTION UNIT

PDU is used to switch on and off all camera electronics remotely. To achieve the controlling and monitoring of PDU the SNMP is used.This is a client-server architecture, where the server is KATCP device, which communicates directly with the PDU device and the client, is the CAM system software. One of the major roles of the server is to serve as a protocol translator; it translates SNMP command messages to KATCP command messages and from KATCP messages to SNMP messages.

HANDLING SNMP

When monitoring the PDU (whether outlets are on or off) the client polls or requests the status of all the outlets. This exercise of polling is impractical and expensive since the client has to poll or request information from every object. This process can be diagrammatically represented in Figure 1. The solution is to use traps. Traps are alerts/notifications generated by the PDU. Figure 2 illustrates the use of traps. Trap messages will be sent to a predefined trap host.

A predefined trap host has snmptrapd (Simple Network Management Protocol Daemon). Snmptrapd is trap daemon running that is waiting for the traps. It receives traps from the PDU at port 162. Snmptrapd has the capability of executing a particular script or command depending on the Object Identifier (OID). Snmptrapd is a linux command line program and is easy to use. After snmptrapd receives the trap it determines which script to run and directly polls the associated device to get a better understanding of the event/trap. Scripts are light and simple KATCP client scripts that receive (read in) traps from snmptrapd and update the CAM system. The script has a mechanism of knowing the destination (host and port) of the trap.

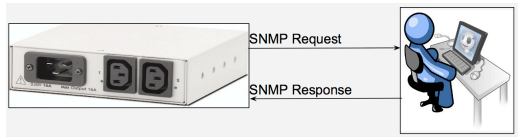

Figure 1: Polling.

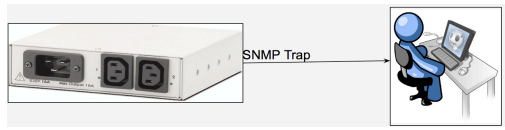

Figure 2: Traps.

START SNMPTRAPD

Starting snmptrapd is managed using the init script named snmpd. Edit the file /etc/default/snmpd to set the option TRAPDRUN to yes.

TRAP DAEMON CONFIGURATION

The daemon has a simple configuration file. One has to setup snmptd to log into syslog by editing the file /etc/snmp/snmptrapd.conf to set the option authCommunity log public. In the configuration file we defined two pieces of information that tells snmptrapd how to direct the incoming traps, the information is the OID of the incoming trap and the location of the traphandler. In our case the following line was added to the configuration file traphandle .1.3.6.1.4.1.318.0.269 python /home/snmp_apc_script.py, whenever an outlet is switched on/off the script snmp_apc_script.py will be executed.

While it is possible to switch on and off outlets from the web, we turn the outlets on and off via SNMP [3]. This makes it possible to switch on and off outlets automatically. Typically SNMP OID of setting ‘on’ outlet number one is (1.3.6.1.4.1.318.1.1.4.4.2.1.3.1.1). The OID of requesting/polling status of outlet number one is (1.3.6.1.4.1.318.1.1.4.2.2.0.1).

STREAMING

To capture video and snapshots from a Camera and stream them through a webserver we use motion. Motion has simple webserver built in. The video stream is in mjpeg format from cameras. It is able to detect if a significant part of the picture has changed, in other words it can detect motion. Motion comes with a configuration file, which needs to be changed to suit user specifications. To start motion process we run motion -n –c motion.conf. From a browser we point to the IP address of where motion is running at and the associated port number.

CONCLUSION

The text based KATCP protocol utilised in the implementation and development of the Camera software

ISBN 978-3-95450-148-9

solution has the advantage of ease of use and debugging as this protocol is widely known and used in the KAT-7 as well as the MeerKAT. Although the PTZ cameras are more applicable and more versatile than the fixed cameras, they do bring a lot of financial constraint on the project. The PTZ camera requires technical maintenance from qualified personnel to keep it running. The image quality on a PTZ camera can be poor in comparison to a fixed unit camera. This is because the CCD sensors on the PTZ cameras have to be smaller to accommodate the gears and motor mechanisms, so the image quality suffers slightly.

REFERENCES

[1] S. Cross et al, "Guidelines for Communication with Devices", SKA SA, July 2012, http://pythonhosted.org/katcp/_downloads/ NRF-KAT7-6.0-IFCE-002-Rev5.pdf

[2] L van den Heever et al, "MeerKAT CAM Design Description, Rev 1", SKA SA, August 2013, http://tinyurl.com/MeerKAT-CAM-Public-Docs

[3] L van den Heever et al, "MeerKAT CAM Requirement Specification, Rev 2", SKA SA, August 2013, http://tinyurl.com/MeerKAT-CAM-Public-Docs

ISBN 978-3-95450-148-9