Fusion Engineering and Design 96–97 (2015) 938–942

Contents lists available at ScienceDirect

Fusion Engineering and Design

journal homepage: www.elsevier.com/locate/fusengdes

ATCA Shelf Manager EPICS device support for ITER CODAC Core System

Bruno Santos[a,*], Paulo F. Carvalho[a], A.P. Rodrigues[a], Bernardo B. Carvalho[a], Jorge Sousa[a], António J.N. Batista[a], Miguel Correia[a], Álvaro M. Combo[a], Nuno Cruz[a], Carlos M.B.A. Correia[b], Bruno Gonçalves[a]

[a] Instituto de Plasmas e Fusão Nuclear, Instituto Superior Técnico, Universidade de Lisboa, 1049-001 Lisboa, Portugal
[b] Centro de Instrumentação, Departamento de Física, Universidade de Coimbra, 3004-516 Coimbra, Portugal

HIGHLIGHTS

- This architecture targets the health management integration into the NDS.
- The developed solution supports the ShM redundancy features, specified by ATCA.
- The average RTT was around 59 ms and in 99.9% of the cases was less than 130 ms.
- Without losing any update cycle, can monitor a system shelf with approximately 400 sensors.
- This solution enables the user to configure the entire system in DB files and *st.cmd*.

ARTICLE INFO

Article history:
Received 5 October 2014
Received in revised form 7 May 2015
Accepted 26 May 2015
Available online 20 June 2015

Keywords:
EPICS
ATCA
Shelf Manager
Control system
ITER
Nuclear fusion

ABSTRACT

The ITER CODAC Core System (CCS) is responsible for plant Instrumentation and Control (I&C) supervising and monitoring. This system uses the Enhanced Physics and Industrial Control System (EPICS) Channel Access (CA) protocol as the interface with the Plant Operation Network (PON).

This paper presents a generic EPICS device support developed for the integration of the ATCA Shelf Manager (ShM) into the ITER CCS, providing scalability and easy configuration. The device support uses the available HTTP interface on Shelf Manager in the communication layer. Both HTTP server and sensors/actuators definitions can be configured using the EPICS database file and the Input/Output Controller (IOC) initialization file. A proposal based on this device is also presented, targeting the Nominal Device Support (NDS) for health management.

The EPICS device support running in an IOC provides Process Variables (PV) to the PON network with the system information and these PVs can be used by all CA clients, such as EPICS user interface clients, alarm systems and archive systems.

Operation with redundant ATCA ShMs and device support scalability tests were performed and the results are presented.

1. Introduction

The Advanced Telecommunications Computing Architecture (ATCA) with extensions for Physics was selected as one of the fast controller instrumentation standards for ITER diagnostics. The ITER Fast Plant System Controller (FPSC) prototype contributed to demonstrate the usability of the ATCA standard and its enhanced specifications for the high-speed and high channel density data acquisition applications. This is required by the most demanding ITER tokamak plasma Instrumentation and Control (I&C) systems [1,2].

The ATCA standard [3] specifies a mandatory Shelf Manager (ShM) unit that monitors the hardware state of the Shelf and is a key element for the system operation. It is responsible for system management and monitoring, as well as for the implementation of high-availability, supporting the use of these systems in long pulse operation. The Intelligent Platform Management Controller (IPMC), mandatory in each intelligent Field Replaceable Unit (FRU), monitors the FRU health, retrieves inventory information and controls the FRU hot-swap operations. The ShM retrieves the data from the

* Corresponding author. Tel.: +351 964514686.
E-mail address: bsantos@ipfn.ist.utl.pt (B. Santos).

http://dx.doi.org/10.1016/j.fusengdes.2015.05.067
0920-3796/

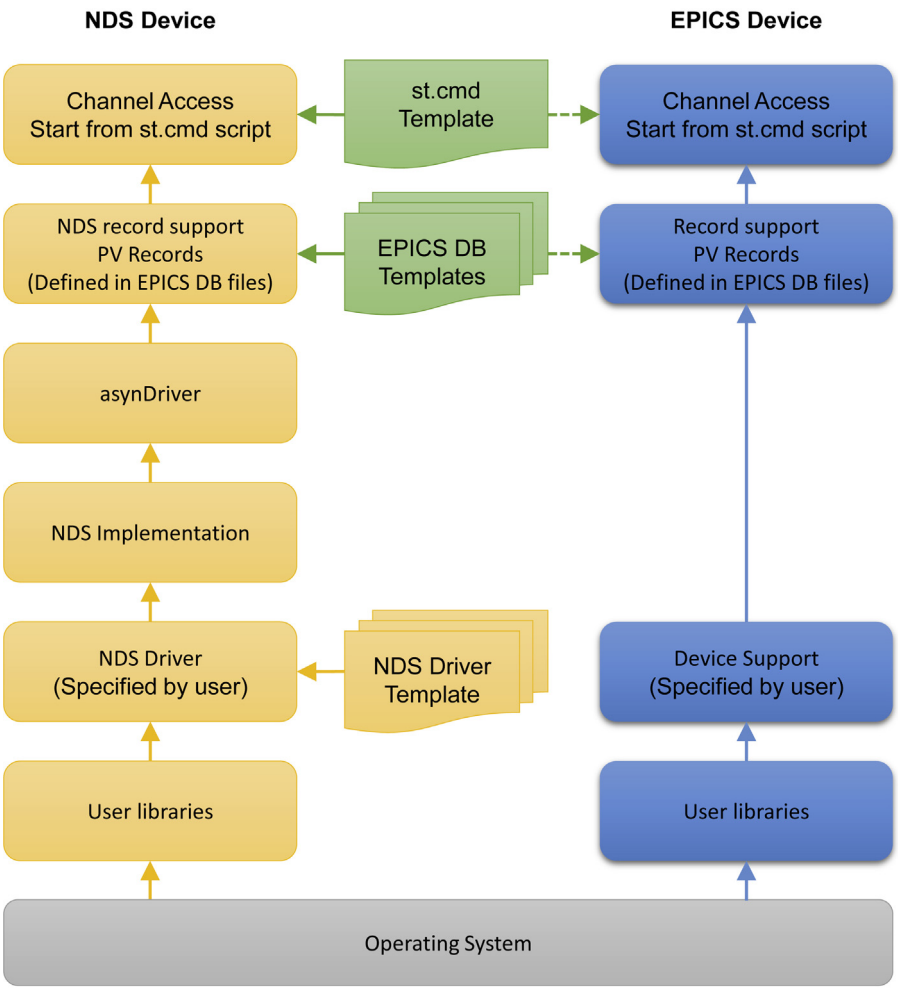

Fig. 1. EPICS and NDS device support.

IPMCs and manages them, ensuring the correct operation of the system [4,5]. Also, the ShM controller can be accessed using a connected terminal or several protocols over Ethernet like RMCP (IPMI over TCP/IP), HTTP or SNMP [6].

The ITER FPSC prototype contains digitizing units (ATCA-IO-PROCESSOR) [7] and PCIe and timing switches (ATCA-PTSW-AMC4) [8] which deliver acquired data to an external host computer. These units also contain an IPMC, making it compliant with ATCA specification.

EPICS is a set of Open Source software tools, libraries and applications developed collaboratively and used worldwide to create distributed soft real-time control systems for scientific instruments [9]. It provides a set of tools for monitoring and configuration and has been selected as the ITER CODAC Core System (CCS) middleware for plant systems I&C [10,11]. The EPICS device support is a software layer which provides the integration of hardware devices, comparable to the abstraction provided by a device driver on an Operating System. The device support registers read/write functions that monitor/configure the devices. These functions are directly connected to the EPICS record configured in the EPICS Database (DB) files which are loaded in the Input/Output Controller (IOC), allowing the IOC Channel Access (CA) server to publish the records on the network as Process Variables (PVs), implementing a distributed control system.

On the client side, there can be many CA clients such as user interfaces clients, alarm systems and archive systems which access the published PVs. The ITER CCS environment includes the Control System Studio (CSS) as Operator Interface (OPI) for showing the system environment in the control room, Best Ever Alarm System Toolkit (BEAST) as alarm system for reporting problems and Best Ever Archive Utility (BEAUTY) that stores data for future analysis.

An HTTP-based EPICS device support for ATCA ShM is presented in [12], however it exhibits communication throttling when the number of sensors increases. That implementation sends the requests using a multi-thread approach which floods the ShM HTTP server exceeding the ShM hardware response capabilities. Therefore, a new software solution based on EPICS callbacks and blocking queues is presented, providing stable communication. Also, new features have been implemented for easy configuration and scalability, enabling the operator to configure the server or add new sensors using the EPICS DB and the IOC initialization file, avoiding changes in device support code. These changes result in a new architecture targeting the health management integration into the Nominal Device Support (NDS) as is laid down by ITER. To implement this new architecture, HTTP was selected among the protocols supported by the ShM, because it is the least complex and complies with all architectural goals. It also allows result comparisons between this solution and the previous version, contributing to the enhancement and demonstration of the new architecture.

2. Software architecture

The NDS is the C++ implementation of the Nominal Device Model (NDM), generalizing the EPICS device support interface for Data

Acquisition (DAQ) and timing devices. It provides a library with a set of interfaces, solutions and best practices to integrate a DAQ or Timing device into EPICS [13,14].

Fig. 1 depicts the architectures of a NDS device support and a pure EPICS device support. For a pure device support the implementation has only one level, which the developer can split in sub-levels or classes according to the best programming practices [15]. To implement a device according to this architecture there is a set of rules to follow but each developer can use his own architecture, which may lead to maintenance and upgrade problems. To avoid these issues and simplify the implementation, the NDS provides base classes which the developer extends with device-specific functionality. NDS also provides templates for device support to make initial development easier. These templates define the interfaces and the most common functions for DAQ devices. Moreover, the NDS inherits the advantages of the *asyndriver* [16] for asynchronous communication, hiding its complexity from the NDS driver developer through the NDS implementation and its templates [14].

Additionally, there are DAQ systems that require other driver template features, such as health management, which are not yet supported by the NDS but are expected to be in the future. The device architecture depicted in Fig. 1 was designed aiming at the integration on the NDS with a minimum of changes. This diagram exhibits some similarities in the upper levels, in which the *st.cmd* and EPICS DB templates are used in both approaches. The usage of templates is a key feature, which provides the entire ATCA health system setup, enabling the users to configure the system in a few steps. In the developed EPICS device, at the driver level, the global concept was to assign each PV to a configuration or status hardware management parameter, such as a temperature sensor value or a board status, depending on the ATCA module. Additionally, the usage of an intermediate level between the PV records and the device support implementation was discussed because the NDS Driver uses the *asyndriver* and it covers a number of standard communication interfaces. In fact, the *asyndriver* can be used because it implements the Ethernet communication used by the ATCA ShM. However, its complexity would burden the internal device support architecture with no advantages. Using C standard libraries the implementation of this type of communication becomes straightforward, provides more flexibility for the implementation and abstracts the developer from the *asyndriver* complexity such as in a NDS driver implementation. Furthermore, the design of a new and generic solution for different types of ATCA ShM clients includes many features that are not provided by the *asyndriver*, such as the support for several protocols and functions to encode/decode the messages. Fig. 2 depicts the device internal architecture, in which the modules to implement this features can be identified.

3. Device support implementation

The purpose of this work is to control and monitor the ATCA FRUs sensors, implementing and testing the architecture depicted in Figs. 1 and 2, aiming at the control of the entire system. HTTP was selected for the communication and protocol decoder layers, providing the connection between EPICS and the ATCA ShM. This is the simplest protocol to implement from those available in the ShM server, which allows a fast and reliable implementation without compromising any of the intended features. In addition, the work carried out on the previous developed version presents a good starting point with room for improvements. This choice was selected bearing in mind that HTTP has several security concerns [17,18], which can be fixed in the future replacing the communication layer by other protocol like SSH or IPMI over LAN.

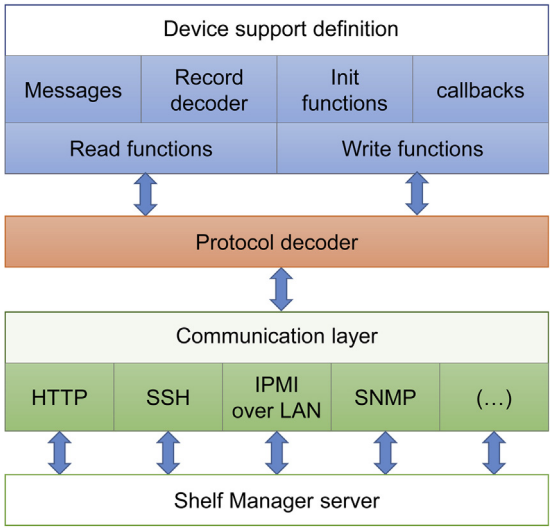

Fig. 2. Device support internal architecture.

This section introduces the three device hierarchical layers (device support, protocol decoder and communication layer) and their modules, finishing with a brief view of the developed templates and the particular features of the device.

3.1. Device support

The upper layer of the device architecture is a common device support implementation which includes the following modules:

- The “messages” module is responsible for the definition of all *error*, *warning* or *information* messages sent to the user and EPICS error logger during the device execution.
- The “init functions” group some initialization functions that are available to provide configurations from *st.cmd* like ATCA modules address, ShM server parameters or other global device configurations.
- The “record decoder” validates and gets the parameters configured in the INP/OUT record fields to identify the configured sensor.
- The “callbacks” module was added because the use of EPICS callback queues is recommended for asynchronous communication which is the characteristic of the several protocols supported by the ATCA ShM.
- The “device support definition” implements the common structures and functions to define each record type, which is directly connected to the write and read functions.

3.2. Protocol decoder

The protocol decoder is the layer responsible for translating the received/sent messages from/to communication protocol format to/from EPICS format. In the case of the HTTP protocol decoder, this layer receives the EPICS parameters from INP/OUT record fields and sends the HTTP message to the server according to these fields, which identify the module id, sensor id, sensor type and also the value to set if is a write PV.

3.3. Communication layer

This layer connects directly to the ShM server and implements the functions to connect, disconnect, send and receive messages which can be configured according to the selected protocol. Since the ShM server supports several protocols like HTTP, SSH, IPMI over

LAN or SNMP, this layer implements the interfaces for each protocol with their specific details. Furthermore, the developer can implement different approaches for the same protocol. For example, to communicate with the HTTP server, it can implement a communication solution using pure C libraries or using other libraries like *curl* [19]. In addition, this layer was configured to provide the communication for persistent and non-persistent connections. However, the persistent connection was implemented for future purposes, since the ShM HTTP server does not support keep-alive connections.

3.4. Templates

The solution includes templates for the *st.cmd* file and for EPICS DB files. The global parameters for the ATCA Shelf are configured in the *st.cmd* template and are used by the EPICS DB templates and device support initialization functions. These include configurations such as IP addresses, port parameters, ATCA module address and macro definitions for the EPICS DB templates.

The EPICS DB templates have similar organization to the ATCA Shelf, which includes these modules:

- Shelf Management Alarm Module (SAM)
- Fan Tray Module (FTM)
- Power Entry Module (PEM)
- Blade (ATCA Front Board) Slots

The installed modules or boards may contain a number of FRUs, which may include sensors to monitor the temperature, power consumption or fan level, depending on the type of FRU. Thus, there is a template for each module which defines its sensors and configurations. Since the board slots can support different board types, some of which (the carrier boards) may contain Advanced Mezzanine Cards (AMCs), there is a template for each board type or AMC which includes its sensors. Additionally, there is a template for the server parameters that can be configured and monitored in runtime.

3.5. ATCA features

The developed implementation supports the ShM redundancy features, specified by ATCA, enabling EPICS to keep track of active ShM changes, with seamless operation for the end user. Also the SAM, FTM and PEM module definitions support redundant operation using the EPICS DB templates.

To support the EPICS connection to the sensors, records can be configured as *ai*, *bi*, *bo*, *mbbi*, *longin*, *longout* and *stringin* types, according to the type of sensor to monitor or configure. Additionally to the sensor values and server settings, the names, slots and status of the installed boards in the system can be monitored from EPICS as well as the possibility to do a board reset from EPICS.

3.6. Mapping to NDS

The NDS internal structure is defined by a class inheritance hierarchy of device, channel groups and channels. Its class objects have a list of functions that can be accessed with the write and read operations of the EPICS records. Moreover, a device can have a number of initialization parameters which can be set at IOC initialization, inside EPICS *st.cmd* start-up scripts [14].

To integrate ATCA health management into NDS, it needs to be mapped in the NDS structure. In the ATCA health management concept, a device corresponds to the ATCA Shelf, a channel group corresponds to each individual ATCA module and a channel corresponds to each sensor. Fig. 3 summarizes this correspondence and depicts a proposal to create the ATCA NDS objects based on the NDS

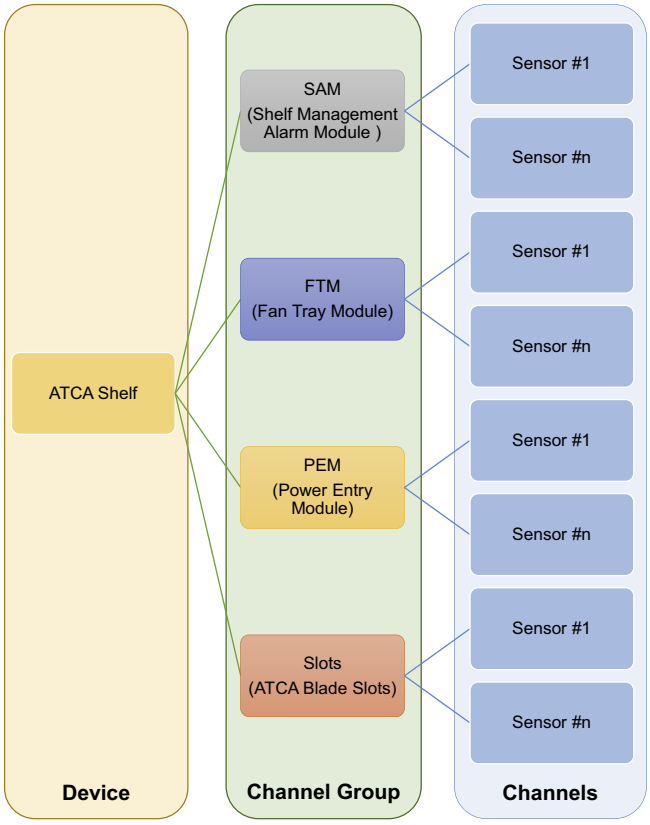

Fig. 3. Heath management for NDS structure.

internal architecture. This structure extends/overrides the available NDS base classes to implement the ATCA health management features.

The NDS driver read and write functions need to be overridden to implement the health management features. For this purpose, the architecture presented in Fig. 2 can be used with the same functions. This structure guarantees a generic solution that allows the addition of other supported protocols by the ShM without changes in the upper levels. However, the device support definition layer has to be adapted to NDS driver template. In the EPICS DB templates, the INP/OUT notation has to be migrated to the *asyn* notation because the NDS uses this notation and the presented solution uses a similar one.

4. Results

Fig. 4 depicts the Round-Trip Time (RTT) histogram for each sensor HTTP request using the developed EPICS device support. The RTT was obtained measuring the time to get the response for

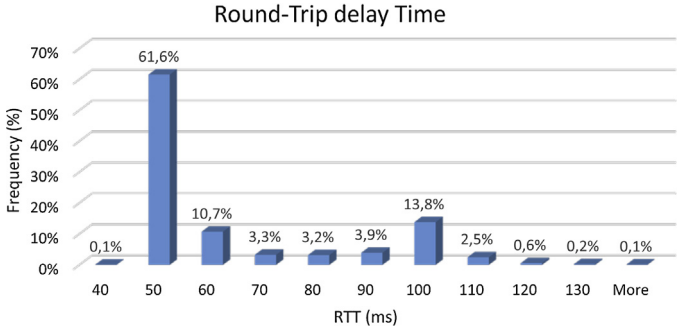

Fig. 4. EPICS device support HTTP communication RTT.

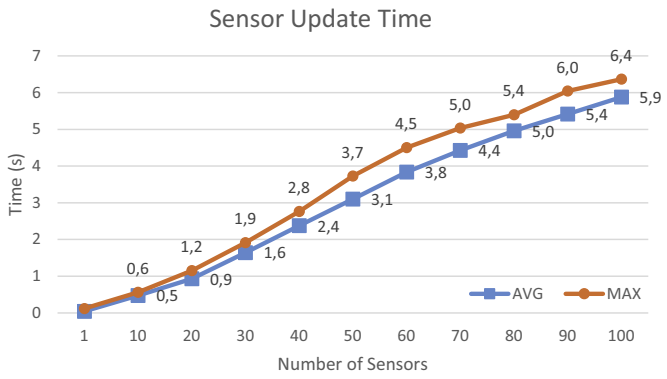

Fig. 5. Sensor update time.

each sensor request. There were half a million samples acquired during approximately 8 h. The results show a RTT minimum of 40 ms and maximum of 988 ms. The average time was around 59 ms with a 21 ms standard deviation. Moreover, in 99.9% of the cases, RTT was less than 130 ms. In addition, the same approach was tested using the *curl* library instead of a pure C implementation. The test result yields a difference less than 1 ms in the average RTT.

Fig. 5 depicts the average time between two successive reads of the same sensor, depending on the number of simultaneously monitored sensors. The EPICS scan time was configured to generate requests on the callback queue immediately after the previous completed action. The results show that this number increases according with the number of sensors nearly 60–70 ms per sensor. This is an expectable value, taking into account the RTT measurements and that the messages are sent sequentially to the ShM server. The difference from the average to the maximum value was between 300 ms and 700 ms, fluctuating according to the monitored sensor type, which has different reply messages with different size.

5. Conclusions

According to these results and taking into account that the ShM updates the sensors value every 30 s, using this approach, the EPICS can monitor a system shelf with approximately 400 sensors without losing any update cycle, within the available ShM CPU performance capabilities.

This solution was designed to permit the user to configure the entire system in DB files and *st.cmd* avoiding code changes. New sensors/actuators to be monitored/operated can be added (or removed) using only the DB files.

The integration into NDS is under development but the present proposal can be a useful case study to improve NDS driver to support hardware management features. Furthermore, the presented device support was developed according to CCS best practices. It is compliant with CCS and can be installed on other Linux version running EPICS.

Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement number 633053. IST activities also received financial support from "Fundação para a Ciência e Tecnologia" through project Pest-OE/SADG/LA0010/2013. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

References

[1] B. Gonçalves, et al., ITER fast plant system controller prototype based on ATCA platform, Fusion Eng. Des. 87 (12) (2012) 2024–2029.
[2] B.B. Carvalho, et al., The ITER fast plant system controller ATCA prototype real-time software architecture, Fusion Eng. Des. 88 (6–8) (2013) 541–546.
[3] PICMG 3.4 document: https://www.picmg.org
[4] A.P. Rodrigues, et al., Intelligent platform management controller for nuclear fusion fast plant system controllers, IEEE Trans. Nucl. Sci. 58 (4) (2011) 1733–1737.
[5] A.P. Rodrigues, et al., Intelligent platform management controller software architecture in ATCA modules for fast control systems, IEEE Trans. Nucl. Sci. 61 (4) (2014) 2318–2322.
[6] M. Overgard, Applying ATCA hardware platform management to IoT backend systems, RTC Mag. (2014 April) 30–33.
[7] A.J.N. Batista, et al., ATCA/AXIe compatible board for fast control and data acquisition in nuclear fusion experiments, Fusion Eng. Des. 87 (12) (2012) 2131–2135.
[8] M. Correia, et al., ATCA-based hardware for control and data acquisition on nuclear fusion fast control plant systems, IEEE Trans. Nucl. Sci. 58 (4) (2011) 1701–1705.
[9] EPICS website, http://www.aps.anl.gov/epics
[10] F. Di Maio, et al., The CODAC software distribution for the ITER plant systems, in: 13th International Conference on Accelerator and Large Experimental Physics Control Systems, 2011.
[11] F. Di Maio, et al., CODAC Core System Overview (ITER 34SDZ5), v 4.3, 2014.
[12] F. Paulo, Carvalho, et al., EPICS device support module as ATCA system manager for the ITER fast plant system controller, Fusion Eng. Des. vol. 88 (2013) 1117–1121.
[13] S. Isaev, et al., Nominal Device Support: Users's Manual Guide (ITER A6LWQ8), v 1.2, 2014.
[14] V. Isaev, et al., EPICS data aquisition device support, in: Proceedings of ICALEPCS, San Francisco, CA, USA, 2013.
[15] EPICS application developer's guide.
[16] asynDriver, asynDriver website: http://www.aps.anl.gov/epics/modules/soft/asyn/
[17] rfc1945 Hypertext Transfer Protocol – HTTP/1.0: https://tools.ietf.org/html/rfc1945#section-12
[18] K. Hemanth, Security problems and their defenses in TCP/IP protocol suite, Int. J. Sci. Res. Publ. (2012) 2–12.
[19] cURL library website, http://curl.haxx.se/