EVALUATING CONTROLLER NETWORK DATA EXTRACTING PROTOCOL FOR EMBEDDED DEVICES

Nikolay R. Kakanakov, Mitko P. Shopov

Department of Computer Systems and Technologies, Technical University of Sofia, branch Plovdiv, 61 "St. Petersburg" Blvd., Plovdiv 4000, Bulgaria, phone: +359 32 659758, www: http://net-lab.tu-plovdiv.bg/ , e-mail: {kakanak, mshopov}@tu-plovdiv.bg

Test-bed experiments of semi-customized application layer protocol for data extraction in distributed embedded systems are presented. Test-bed network and configuration are built in "Laboratory for Computer Networks and Distributed Systems" in Technical University, branch Plovdiv. The presented experimental results provide a base data for evaluation the protocol leaks and performance. Additionally, the experiments are executed on two different embedded platforms, running different implementation of the protocol – in Java and C. It allows separation of platform specific components in the experimental results. The experiments are monitored and analyzed using specially built application. The results are stored in XML files for further analyses.

Keywords: Distributed Embedded Systems, Remote Monitoring and Control.

1. INTRODUCTION

Over the recent years there is a trend for adaptation of emerging computer technologies in Distributed Embedded Systems. The vendor specific standards are replaced with popular open standards for communication between embedded devices (TCP/IP, Ethernet, WiFi). The embedded communication is being integrated with Internet paradigms and Internet-ready embedded devices appear on the market. The popular enterprise technologies and protocols cannot be directly applied to automation systems and home appliances. The primary recognized problems are the resource limitation of the embedded devices and the need of fast response to changes in the environment. A solution to these problems can be a semi-customized application protocol for interaction with the embedded systems, built upon standard communication stacks (TCP/IP). Such protocol should be a trade-off between universal and application specific realizations. The fully-customized protocols will hardly be adapted to different platforms and standard protocols are too resource consuming to deploy on embedded devices [5, 7].

2. PROBLEM STATEMENT

As long as a new protocol is designed and implemented, it should be evaluated and tested. This includes tests against the requirements form the environment and evaluation of main characteristics of the protocol (packet's size, packet distribution, protocol delay). The paper presents test-bed experiments for evaluation of Controller Network Data Extracting Protocol (CNDEP). The experimental data collected during the test-bed will be used for:

- addressing the leaks in the protocol implementation;
- evaluating the delay that the protocol carries in the communication;
- collecting values for further simulation analysis of more complex systems, based on CNDEP communication.

2.1. Controller Network Data Extracting Protocol (CNDEP)

CNDEP is an asymmetric protocol used in Client/Server systems to "extract" data from microcontrollers working in LAN. It works on the application layer of the TCP/IP stack. As a transport protocol it uses UDP. An area where UDP is especially useful is the client-server applications – this is the CNDEP base. The client sends a short request to the server and expects a short reply back. Current implementation of the protocol is supposed to work in Local Area Network and to be integrated in multi-tier systems for home and office automation. The reasons for using UDP are: short message exchange; fast communication; good reusability of communication channel; service of devices in sensible time. The protocol implements a number of commands – e.g., 'Test', 'GetTemperature', 'GetHumidity', 'SetTemperatureOptions', 'SetHumidityOptions' [2, 4].

2.2. Test-bed architecture and configuration

A test-bed experiments configuration typically consists of experimental subsystem, monitoring subsystem and simulation-stimulation subsystem. The experimental subsystem is the part which characteristics should be fetched. Thus, the simulation-stimulation subsystem provides parameterized inputs to the experimental subsystem and the monitoring subsystem collects the outputs [1, 6].

For executing the experiments an experimental network is built. It consists of several embedded devices and monitoring stations – desktop PC and PDA, interconnected in switched LAN. The architecture of the network for the test-bed is presented on fig. 1. The embedded devices included in the experiments are DS TINI from Dallas Semiconductors [9] and IPC@Chip SC12 from Beck IPC [8].

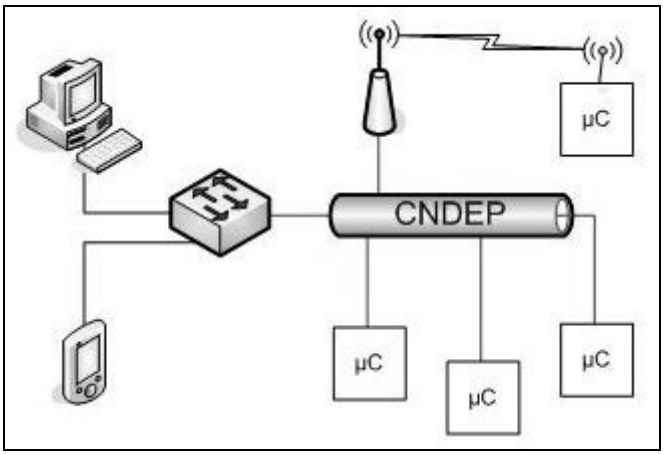

Figure 1: Experimental network.

The configuration of the system-under-test for the test-bed experiments is shown on fig. 2. It is built to address the main requirements of the test-bed experiments, including: network delay times, TCP/IP stack delay times and CNDEP delay times. Experimental data is collected on a data capturing computer.

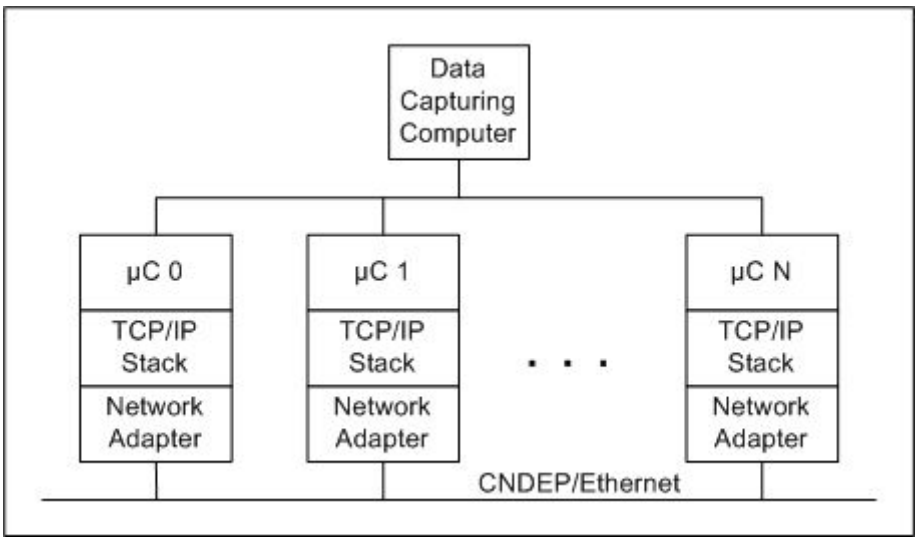

Figure 2: Test-bed configuration

The data capturing computer acts as a CNDEP client and executes requests to controllers under test and receives the responses. Thus, it provides platform for simulation-stimulation subsystem and monitoring subsystem. They are implemented together in a software tool, which tool allows choosing different experimental scenarios and collects the results. The calculated delay, together with the request and response types and corresponding controller info are stored in XML files. The tool provides an interface for choosing the request type (CNDEP command), the number of times for executing the request, and the interval between requests. The time intervals are measured using "HighPerformanceTimer", provided by the Windows framework. It has accuracy of about 0.001ms.

2.3. Target embedded platforms

Target embedded platforms included in the test-bed are DS TINI from Dallas Semiconductors [9] and IPC@Chip SC12 from Beck IPC [8]. On the embedded platforms involved in the experiments runs the server side of CNDEP. For providing the interaction of the device with the environment is used temperature and humidity measurement using SHT71 intelligent sensor [10]. The DS TINI platform uses Java virtual machine for running applications and IPC@Chip runs specific RTOS. In this way the calculated delay times for the protocol are dependable on the platform architecture. Thus, the results can be used for evaluation of the platform themselves.

3. EXPERIMENTAL RESULTS

The evaluation of the protocol implementation includes measuring the latency of networking hardware, the latency in embedded devices for creating a socket and reading/writing to it, and the delay of the protocol itself. All experiments are executed 101 times in relatively big intervals, for providing good statistical results for approximation [1, 6]

3.1. Comparison of communication latencies of CNDEP implementation for the two embedded platforms

Experiments include four different CNDEP commands executed on the both embedded platforms. The commands are 'Test', 'GetTemperature', 'GetHumidity', 'SetTemperatureOptions' [4].

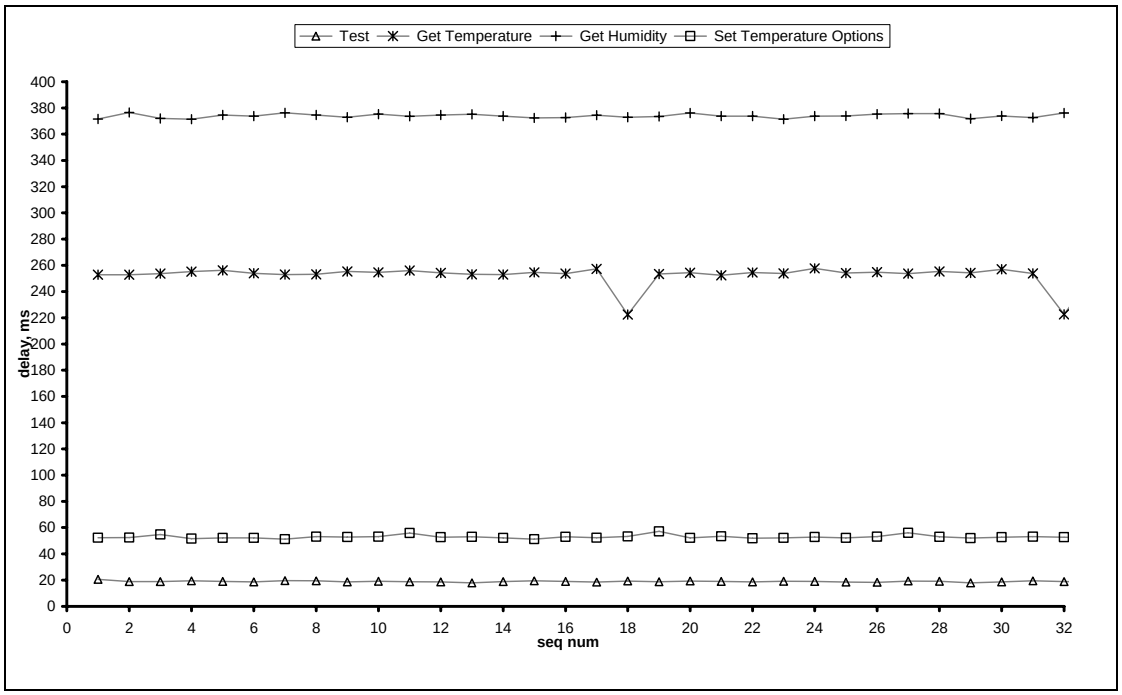

Figure 3: Delay times for CNDEP commands for DS TINI platform.

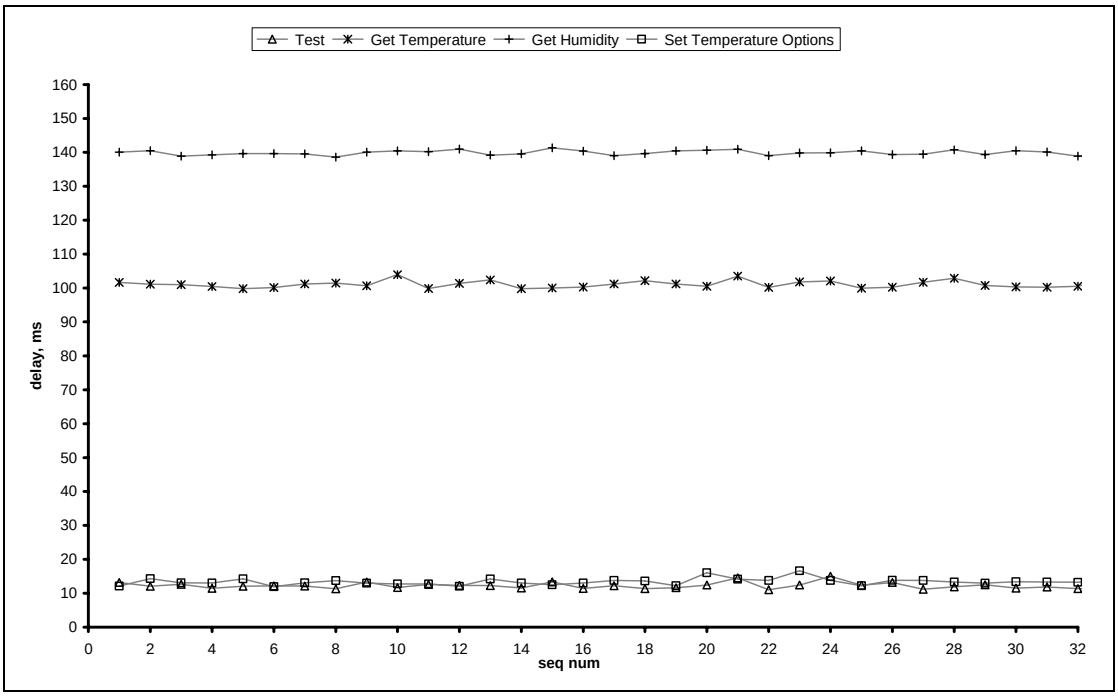

Figure 4: Delay times for CNDEP commands for IPC@Chip platform.

The command delay times for DS TINI is shown on fig. 3 and for IPC@Chip – fig. 4. The presented results are for low resolution sensor measurement. 'Test' command only returns 'OK' response to the request without any other processing on the embedded platform. The other commands demonstrate the Get and Set type of communication. The two Get commands include the latency of the sensor

measurement. The 'GetHumidity' forces the sensor to execute both temperature and humidity measurement for calculating the humidity with temperature compensation.

3.2. Comparison of CNDEP for two platforms, for two different measurement resolutions

For addressing the sensor measurement latency, experiments are executed for high and low resolution measurement for two platforms (results are shown on fig. 5). High resolution is 14bit for temperature (0.02%) and 12bit for humidity (0.03%). The low resolution is 12bit temperature (0.07%) and 8bit humidity (0.5%). The minimum latencies are respectively 210/55/11 ms for 14/12/8 bit measurements, as specified by the vendor [10].

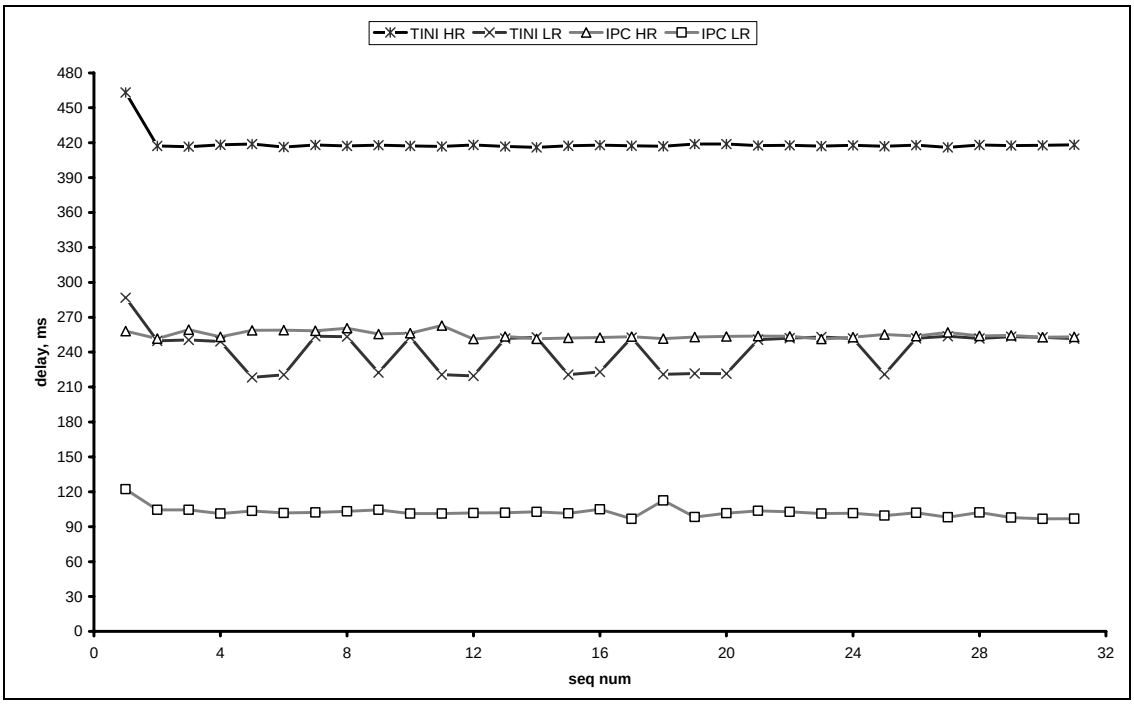

Figure 5: Comparison of 'GetTemperature' command for both devices, high and low resolution measurements (HR and LR).

Using low resolution, not only the sensor works faster, but the power consumption is smaller. It is useful for real-time or power-saving applications, without the need of strict accuracy. The minimum, maximum and average values of three types of command, together with their deviation are shown on table 1.

Table 1

command / controller		TEST			GET			SET		
		min	**max**	**avg ± σ**	**min**	**max**	**avg ± σ**	**min**	**max**	**avg ± σ**
IPC@Chip	**HR**	10.95	14.5	13.4 ± 1.36	251.18	262.77	254.7 ± 0.17	11.89	16.41	14.1 ± 0.67
	LR	-	-	-	96.66	122.21	102.4 ± 0.27	-	-	-
DS TINI	**HR**	17.38	21.59	19.1 ± 0.09	415.89	463.26	418.9 ± 0.45	51.16	57.08	52.8 ± 0.12
	LR	-	-	-	218.27	286.81	242.1 ± 0.93	-	-	-

4. CONCLUSIONS

The results of the test-bed experiments show that CNDEP implementation is fast and response times has no significant deviation from its mean value. Thus, CNDEP communication shows to be predictable for the environment it is designed for – local

network of controllers. The experiments also provide data for comparison between communication latencies of the embedded platforms – DS TINI and IPC@Chip.

The CNDEP command latencies for both platforms can be compared to the latencies of a simple UDP 'echo' server, provided in [3]. Thus, the latency of the protocol can be separated form the delay of the communication media and latency of the TCP/IP stacks. For IPC@Chip, the 'Test' command is executed for about 13.5ms, the 'SET' – 14.1ms, and simple 'echo' – 9.3ms. The calculated latency of the protocol is about 4.2ms. For DS TINI, the 'Test' command is executed for about 19.1ms, the 'SET' – 52.8ms, and simple 'echo' – 14.8ms. The calculated latency of the protocol is about 4.3ms. The CNDEP 'Test' command shows that CNDEP message parsing takes approximately the same time for both controllers. On the other hand 'SET' commands have significant difference for the both controllers. This can be explained by the need of context switching between Java virtual machine and Native Interface for DS TINI.

5. FUTURE WORK

The test-bed architecture includes only two popular embedded platforms and future work will be directed to implementation and evaluation the CNDEP for additional platforms with different architectures. Another direction in the future work is using the results of the test-bed for creating scenarios for simulation of the CNDEP-based network of controllers for wired and wireless implementations.

6. ACKNOWLEDGEMENT

The presented work is supported by National Science Fund of Bulgaria project – "**ВУ-966/2005**", entitled "Web Services and Data Integration in Distributed Automation and Information Systems in Internet Environment", under the contract "**ВУ-МИ-108/2005**".

7. REFERENCES

[1] Fortier, P., G. Desrochers, *Modeling and Analysis of Local Area Networks*, CRC Press, 1990.

[2] Holzmann, G., *Design and Validation of Computer Protocols*, Prentice Hall, 1991, ISBN: 0-13-539925-4.

[3] Kakanakov, N., *Experimental analysis of client/server applications in embedded systems*, Proc. ELECTRONICS-ET, Sozopol, September 2005, book 4, pp. 97-102, ISBN:954-438-520-7.

[4] Kakanakov, N., I. Stankov, M. Shopov, and G. Spasov, *Controller Network Data Extracting Protocol – design and implementation*, Proc. CompSysTech'06, Veliko Tarnovo, June 2006, pp. IIIA-14.1 – IIIA-14.6.

[5] Sridhar, T., *Designing Embedded Communications Software*, CMP Books, 2003, ISBN: 1-578-20125-X.

[6] Stallings, W., *High-Speed Networks and Internets*, 2nd Ed, Prentice Hall, 2002, ISBN: 0-13-032221-0.

[7] Youngblood, G. M., *Smart Environments*, Ch. 5: "*Middleware*", pp. 101-127, 2004, ISBN: 0-471-54448-5.

[8] http://www.beck-ipc.com/ipc/ - IPC@Chip Homepage [July 2006].

[9] http://www.maxim-ic.com/products/tini/ - DS TINI Homepage [July 2006].

[10] http://www.sensirion.com/en/02_sensors/, Sensirion Inc., "Humidity Sensor SHT71", [July 2006].