Round-Trip Delay Estimation in OPC UA Server-Client Communication Channel

Zilvinas Nakutis[1], Vytautas Deksnys[1], Ignas Jarusevicius[1], Vilius Dambrauskas[1], Gediminas Cincikas[1], Alenas Kriauceliunas[1]
[1]*Department of Electronics Engineering, Faculty of Electrical and Electronics Engineering, Kaunas University of Technology,*
Studentu St. 50–443, LT-51368 Kaunas, Lithuania
zilvinas.nakutis@ktu.lt

***Abstract*—In this paper an estimation of round-trip delay (RTD) in OPC UA server-client channel was investigated in various data communication networks including Ethernet, WiFi, and 3G. Testing was carried out using the developed IoT gateway device running OPC UA server and remote computer running OPC UA client. The server and the client machines were configured to operate in Virtual Private Network powered by OpenVPN. Experimental analysis revealed that RTD values are distributed in the wide range exhibiting difficult-to-explain outliers significantly exceeding average RTD value. A preliminary exploration of the correlation between instantaneous load of communication gateway processor and RTD peaks was carried out on ARM Cortex A8 Texas instruments processors running at 600 MHz and 800 MHz clock frequency.**

***Index Terms*—Round-trip delay; server-client communication; processor load.**

I. INTRODUCTION

Merge of automation networks and data communication networks takes place in the Internet-of-Things (IoT) architectures. An important role in the IoT architecture is dedicated to a so called IoT communication gateways (CGW), that are aimed to ensure connectivity between heterogeneous field networks (industrial buses, wireless, machinery, building automation, sensors networks, etc.) and IP based wide area data communication networks. Some of the described IoT gateway architectures are dedicated to the conversion between two connectivity protocols, for example Zigbee to TCP/IP [1], CAN to GPRS [2], CAN to TCP/IP (WiMAX) [3], while other IoT gateways are multiprotocol devices especially at the field side [4], [5].

A CGW exposes access to field data sinks and sources towards Cloud hosted services. Mostly two technologies Simple Object Access Protocol (SOAP) and Representational State Transfer (REST) are discussed for web services provision [5], [6]. As a standard for application level protocol the OPC UA standard (Open Platform Communication Unified Automation) (https://opcfoundation.org/) is expected to play a noticeable role in IoT [7]. OPC UA protocol was designed for machine-to-machine industrial interoperability and now is well established in industrial automation field. OPC UA is also a cross-platform SOA (Service Oriented Architecture) for process control.

Compared to the REST approach, SOAP style OPC UA features a standardized security architecture, availability of mature commercial software development kits (SDK), data modelling tools (address space), and wide acceptance in the field of industrial automation. Advantages of REST approach are easier implementation, less requirements on the performance of embedded system (resource constrained). These advantages could be crucial for resources limited IoT nodes like remote sensors, but are less critical to infrastructure devices like CGW. Recently described CGW implementations are based on quite powerful processors like ARM 9 [1], [8]–[9], ARM Cortex M4 [10], Intel Atom/Quark SoC processors [11], etc.

The goal of our research is to investigate achievable round-trip delay (RTD) between OPC UA server and client in different data communication networks like Ethernet, WiFi, 3G. Range of RTD parameter is needed to identify possible application areas and characteristics of processes that may be monitored or controlled using CGW connected to Cloud services by means of OPC UA standard at application level.

II. RELATED WORKS

Performance of OPC UA server-client communication channel is investigated by Salvatore Cavalieri, *et al*. [12], [13]. Authors used round-trip delay and delay/sampling interval to quantify the performance of OPC UA. Factors influencing the channel performance are:

1. OPC UA protocol and efficiency of its server and client software implementation,
2. OPC UA server and client settings,
3. Underlying networks and used protocols,
4. Computing performance of target systems hardware running server and client applications.

Cavalieri, *et al*. [13] focused on OPC UA performance estimation using simulation environment, this way abstracting from implementation of server and client SDK, target hardware, underlying communication networks (transport layer), etc. The main finding of their research is that OPC UA subscription profile delay/sampling interval and bandwidth utilization is highly dependent up on settings

Manuscript received 18 December, 2015; accepted 10 April, 2016.
This research was funded by European Community's FP7 programme under grant agreement No.604659 (project CLAFIS).

of subscription service (mainly Publish Interval). In our experimental performance testing we run developed software on the CGW and communicate data over a real communication infrastructure. Specification of the test environment in this case is much more challenging due to big number of factors influencing status of communication network load (3G, WiFi, Ethernet). However, physical experiments enable us to produce real estimates of the channel throughput and to verify the influence of CGW hardware and software implementations.

III. COMMUNICATION GATEWAY ARCHITECTURE

We have implemented CGW based on Variscite System-on-Module (SoM) with Linux Debian operating system. Variscite SoM contains Texas instruments Cortex-A8 processor running at 800 MHz clock frequency, 512 MB Flash memory and 512 MB DDR3 SDRAM, and on board WiFi communication module supporting access point (hot spot) and client modes.

At the software level CGW runs OPC UA server, which was developed using Softing GmbH OPC UA SDK for Linux. OPC UA client was developed using Softing GmbH OPC UA client for Windows SDK. OPC UA binary mode is used in all testings presented in Chapter V.

Virtual private network (VPN) was established in order to assign non-global IPv4 address to CGW. For this purpose, OpenVPN clients were installed on both server (CGW) and client workstation. OpenVPN server was configured on the Linux machine connected to Internet through Gigabit Ethernet. VPN was also responsible for channel securing. In order not to duplicate channel data encoding, OPC UA data encryption was disabled.

Figure 1 presents a diagram describing interconnection of OPC UA server and Linux processes denoted by Protocol adapters and responsible for accessing data in field networks. The communication between them is implemented using Linux inter-process sockets technique.

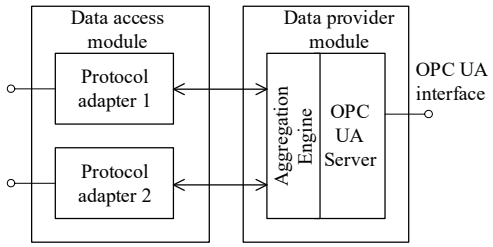

Fig. 1. Simplified component diagram of CGW application level software architecture.

For testing purpose OPC UA server address space containing 30 nodes of double type was created. A node value is updated by simulated data from Protocol adapter every 10 ms in order to simulate data generated by a process under monitoring. It should also be noted that every node in OPC UA address space is automatically supplemented by two time stamps (source and server) each of 8 bytes size.

IV. TEST METHODOLOGY AND ENVIRONMENTS

Round-trip delay t_R is defined as a time interval between moments of data request by client and response reception by the same client.

Test environments comprising the CGW, OpenVPN server, IP gateway, and remote workstation running OPC UA client were setup as specified in Table I and shown in Fig. 2–Fig. 4. The presented network configurations are very common for connecting IoT gateways (like CGW) to TCP/IP Wide Area Networks (WAN). Being connected to WAN the CGW can access Cloud services, for example, to store collected data in SQL or BigData databases. The developed CGW prototype was initially dedicated to facilitate connectivity between precision agriculture sensors or machinery and Cloud services. However, IoT gateway is a general purpose infrastructure device, which can be used in variety of applications in the areas of smart cities, building automation, healthcare, smart grid, etc. The CGW maintains Ethernet, WiFi and 3G communication channels for connectivity towards WAN. The motivation of VPN utilization was already disclosed in Chapter III.

TABLE I. TEST ENVIRONMENTS SPECIFICATION.

Test environment	Description
Ethernet	CGW and workstation running OPC UA client are directly connected to single IP router (ASUS RT-N53).
WiFi-AP	The workstation is connected directly to the WiFi AP (access point) hosted by the CGW. The AP is configured to operate in IEEE 802.11g mode and is secured using Wi-Fi Protected Access II (WPA2) security protocol.
Ethernet-VPN (Fig. 2)	CGW and client workstation are connected to the internet via wired Ethernet connections. They both have OpenVPN clients installed and are using VPN to communicate. The cryptographic functions used by OpenVPN were default (symmetric cypher: BlowFish CBC 128; asymmetric cypher: RSA 1024; message digest: SHA1). TCP/IP stack tool *traceroute* reported 2 hops (routers) to VPN server.
WiFi-VPN (Fig. 3)	CGW is connected to external WiFi AP (ASUS RT-N53). CGW and client workstation are connected through Open VPN tunnel.
3G-VPN (Fig. 4)	Same as the Ethernet-VPN connection option, except the CGW is connected to internet using 3G modem. The traceroute tool reported 9 hops to VPN server.

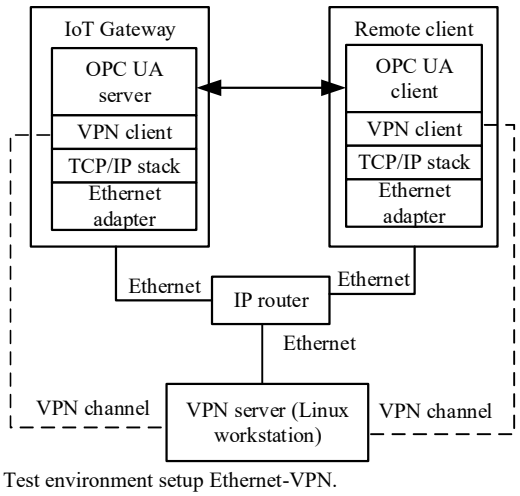

Fig. 2. Test environment setup Ethernet-VPN.

At the application level a remote client sends requests to CGW OPC UA server to retrieve node values according to the selected test case scenario:

1. 1 double type node request at max speed,
2. 10 double type node request at max speed,
3. 30 double type node request at max speed,
4. Same as C except 5 s waiting delays between adjacent response and request were introduced by the client.

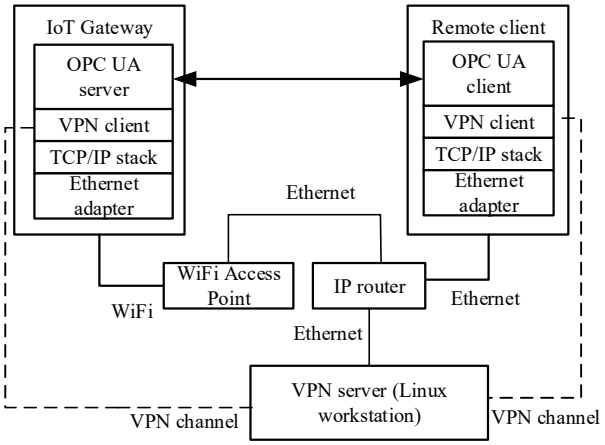

Fig. 3. Test environment setup WiFi-VPN.

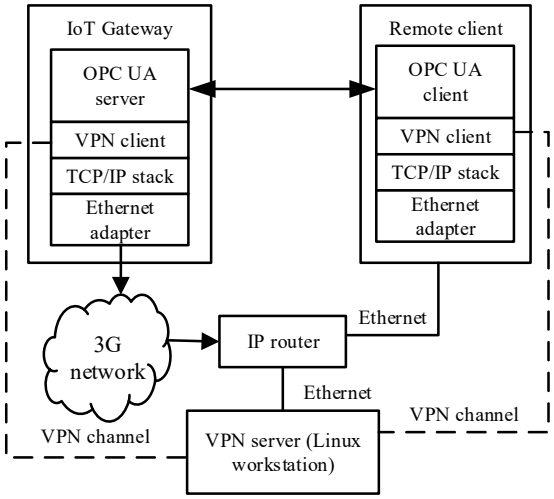

Fig. 4. Test environment setup 3G-VPN.

We have opted such scenarios in order to investigate both single in field acquired quantity delivery to remote client and also a middle size set of field quantities, for example read from machinery buses. To collect samples of RTD for its statistical estimation, 1000 requests were generated by the client. RTD was measured by the client software using Windows API QueryPerformanceCounter function, which retrieves the current value of the target system's high resolution (1 μs) time stamp.

V. ROUND-TRIP DELAY ESTIMATION RESULTS

Round-trip delay average value $E(t_R)$, standard deviation $s(t_R)$, minimal $t_{R\min}$ and maximum $t_{R\max}$ values are calculated from experimentally acquired RTD samples and are presented in Table II.

Throughput of the useful payload can be defined by

$$\mu_1 = \frac{b \times n}{t_R}, \quad (1)$$

where b is number of bits per one OPC UA node value, n is number of OPC UA nodes requested by client and delivered by server (read operation), and t_R is request RTD. The useful payload consists of process parameter values (node values) and their acquisition moment time stamp. For double type nodes 64 bits are used to hold value and 64 bits are used for time stamp. The useful payload throughput is less than the network throughput due to the data overhead of OPC UA protocol. Average value of the throughput $E(\mu_1)$ is calculated from average RTD value $E(t_R)$ using (1), and presented in Table II.

TABLE II. RTD ESTIMATION RESULTS.

Test env.	Scena-rio	$E(t_R)$	$s(t_R)$	$t_{R\min}$	$t_{R\max}$	$E(\mu_1)$
		Units				
		ms	ms	ms	ms	kbps
Ethernet	A	3.0	0.7	2.4	12.0	42
	B	4.4	1.1	3.4	17.1	292
	C	6.7	1.6	5.9	24.7	572
	D	8.8	3.2	4.0	45.0	435
Ethernet-VPN	A	5.0	5.0	3.3	73.2	26
	B	4.4	2.7	3.2	52.4	289
	C	7.3	6.4	4.9	101.0	527
	D	15.6	8.6	5.6	68.0	245
WiFi-AP	A	5.6	6.9	3.4	77.7	23
	B	7.3	14.6	3.2	114.4	175
	C	7.4	9.0	4.9	139.9	520
	D	22.0	62.4	6.2	1663.7	175
WiFi-VPN	A	9.5	12.4	4.9	174.2	13
	B	14.2	18.5	6.7	263.8	90
	C	17.4	29.6	7.1	235.6	221
	D	20.9	22.4	10.1	332.4	184
3G-VPN	A	261.7	134.7	144.4	1141.3	0.5
	B	267.9	139.2	150.2	1705.5	5
	C	250.4	125.2	156.2	1552.7	15
	D	284.9	131.8	162.2	3482.8	13

In Fig. 5 and Fig. 6 histograms of RTD samples are shown. It is interesting to note that in case of 3G-VPN environment (Fig. 5) RTD increases, when 5 s delays are introduced between requests. This feature might be important to consider when remote field process events must be detected fast. In opposite, when only data logging is required, RTD is of less importance because data may be buffered in OPC UA server's address space together with precise time stamp of acquisition moment. Then buffered arrays of data can be delivered to OPC UA client for non-real time inspection.

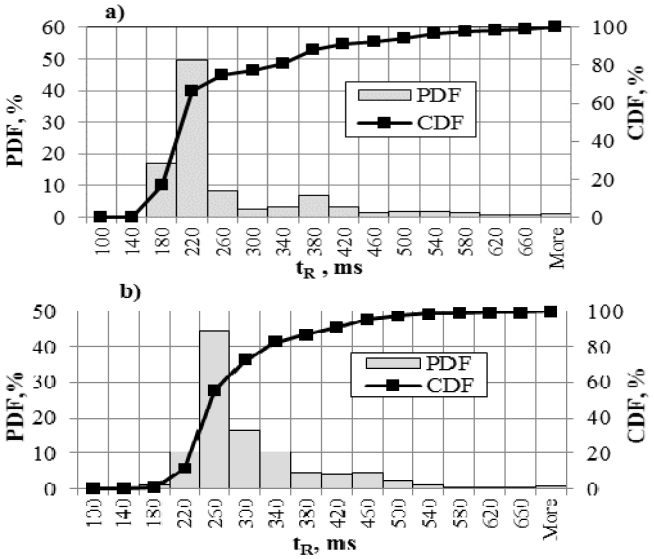

Fig. 5. RTD histograms derived from 3G-VPN test case C scenario (a) and D scenario (b). PDF – probability distribution function, CDF – cumulative distribution function.

From Fig. 6 it is seen that in case of WiFi-VPN environment the probability distribution function (PDF) is

close to exponential and contains a considerable amount of very high RTD values (column "More"). This phenomenon was also observed in all the rest test environments reported in Table II, except scenario D. Because these outliers of RTD can heavily affect the worst case data delivery delay from the observed process, it is of interest to identify the reason of their appearance. Firstly, we examined test environments using ping utility from TCP/IP stack. Delays reported by the ping utility did not indicate presence of any considerable outliers of ping packets RTD. Secondly, we raised a hypothesis that random RTD increases are due to insufficient CGW CPU performance. In chapter VI we give some preliminary results of CPU load observations.

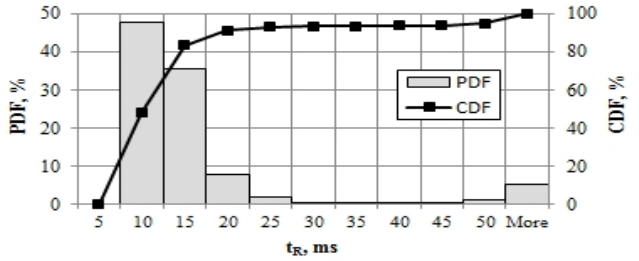

Fig. 6. RTD histograms derived from WiFi-VPN test case C scenario.

It also should be noted that Linux scheduler allocates 10 ms time slots for each active software process (thread) execution. Therefore, every RTD value fluctuation in the range of 10 ms can be due to process switching activities in CGW operating system.

VI. Round-Trip Delay and CPU Load Relationship

To investigate a relationship between server's CPU load and OPC UA channel's RTD, the recording of CPU load was carried out during the testing. Results presented in this chapter were acquired using reduced clock frequency CGW processor (600 MHz instead of 800 MHz), in order to investigate the suspected insufficiency of the processing performance. A sample plots of obtained results are presented in Fig. 7.

Though the recorded RTD and CPU load patterns intuitively seem to correlate, the calculation of Pearson correlation indicated rather weak relationship. CPU load fluctuates due to the various activities of internal Linux processes. We assume that until a certain level the CPU load influence upon RTD is minor. Therefore, we defined the following hypothesis: RTD peaks exceeding the least observed RTD value by N times, are related to the events of CPU load exceeding threshold level K. In order to verify this hypothesis the experimentally acquired RTD and CPU load patterns (Fig. 7(a) and Fig. 7(b)) were transformed to the corresponding binary sequences:

$$t_{Rpeak}(i)=\begin{cases}1, & t_R(i)\geq N\cdot\min(t_R),\\ 0, & t_R(i)<N\cdot\min(t_R).\end{cases} \quad (2)$$

$$L_{CPUpeak}(i)=\begin{cases}1, & L_{CPU}(i)\geq K \text{ and } t_{Rpeak}(i)=1,\\ 0, & \text{otherwise}.\end{cases} \quad (3)$$

Then the Pearson correlation coefficient between these sequences was calculated and presented in Table III (PEA columns). In all cases (N and K value combinations) the significance level (p-value) of hypothesis of no relationship between sequences is less than 0.05, indicating that the corresponding correlation is significant. Similarity between binary patterns alternatively can be estimated using variety of measures [14]. We have selected to calculate Kulczynski-II index (KUL columns). This index represents conditional probability that the peak is present in RTD patterns, given that the peak is also present in CPU load pattern. It can be seen from Table III, that Kulczynski-II index is quite close to Pearson correlation coefficient. Therefore, we assume that the larger are appearing peaks of RTD pattern the higher is probability they are due to the increased load of CPU.

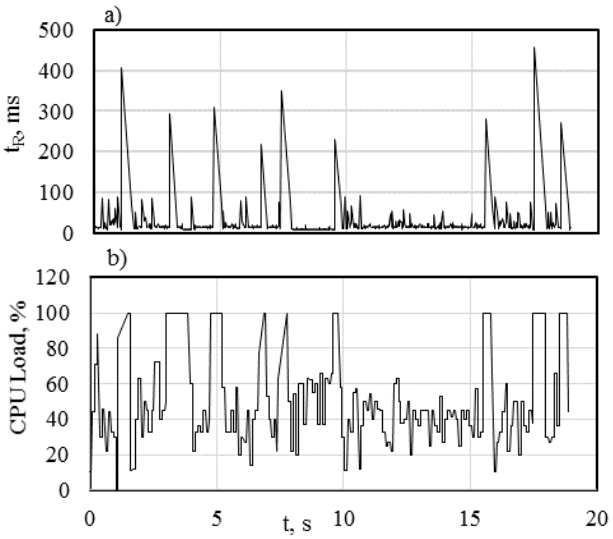

Fig. 7. RTD (a), and CPU load (b) of WiFi-VPN test case C scenario.

TABLE III. RTD PEAKS AND CPU LOAD CORRELATION.

RTD peak threshold gain N (Eq. (2))	CPU load threshold K					
	40 %		50 %		60 %	
	PEA	KUL	PEA	KUL	PEA	KUL
5 times	0.79	0.82	0.70	0.75	0.65	0.72
10 times	0.89	0.90	0.81	0.83	0.75	0.79
20 times	1.00	1.00	1.00	1.00	1.00	1.00

VII. Profiling of CPU Load

The approximate profiling of CGW CPU load due to separate software processes was carried out by reading Linux kernel system statistics files proc/stat (http://man7.org/linux/man-pages/man5/proc.5.html), that provide an ongoing look at processor activity in operating system. The CGW software then added the corresponding readings to its OPC UA server's address space. In particular, load of CPU due to Linux process executing OPC UA Server, Protocol adapter, VPN client, WiFi driver and total CPU load were recorded during the testing. From a typical profiling charts (Fig. 8 and Fig. 9) it is evident that OPC UA server process contributes most significantly to the overall CPU load. Other processes like Protocol adapter, VPN client, WiFi driver are less demanding for processor performance. From Fig. 9 we may see that processing performed by OPC UA server constitute nearly the whole CPU load in Ethernet-VPN environment. In case of the

WiFi-VPN setup some processor resources are occupied by WiFi driver. Therefore, during WiFi-VPN setup testing RTD was larger (see Table II), resulting in less requests arriving from the client per time unit. That means OPC UA server handled less requests in WiFi-VPN case compared to Ethernet-VPN. The whole CPU load was more sporadic in WiFi-VPN case (Fig. 8), perhaps due to WiFi driver execution by the CGW main processor.

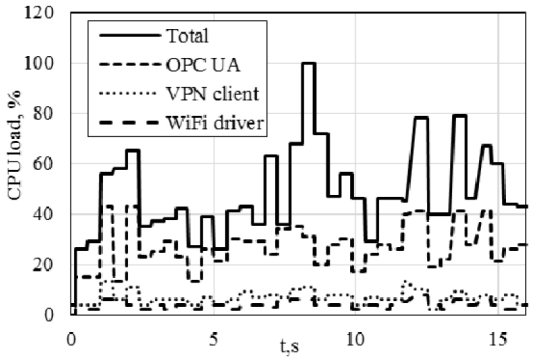

Fig. 8. CPU load by CGW software processes profile during WiFi-VPN test case C scenario testing.

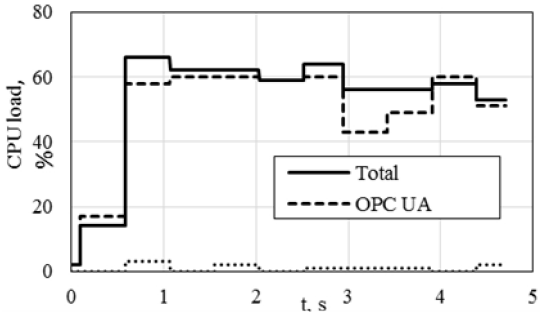

Fig. 9. CPU load by CGW software processes profile during Ethernet-VPN test case C scenario testing.

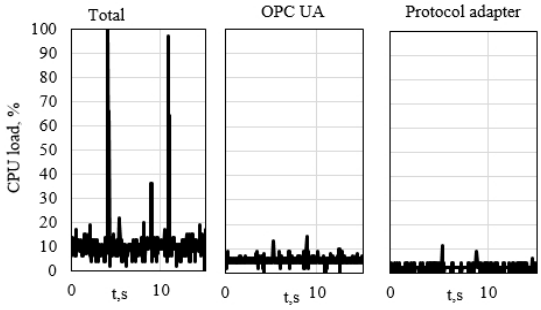

Fig. 10. CPU load by CGW software processes profile during WiFi-VPN test case D scenario except delays between requests were 150 ms testing.

Load of CGW CPU decreases significantly (see Fig. 10), when 150 ms delay is introduced between client requests. Demand for such an update rate (compared to full speed interrogation) is probable for example in applications targeting machinery generated data retrieval from CAN bus and delivery to OPC UA client.

A more thorough investigation and Linux processes profiling is required to identify reasons of RTD outliers appearance in OPC UA data delivery channel.

VIII. CONCLUSIONS

In WiFi and Ethernet based networks OPC UA channel average round-trip delay achievable using 800 MHz ARM Cortex A8 processor running Linux operating system is in the range of several milliseconds per one OPC UA node. Grouping OPC UA nodes is beneficial, because RTD increases less times than the number of nodes is increased (Table II). In 3G network average RTD of OPC UA channel was around 250 milliseconds and little dependent on the number of nodes delivered to client (from 1 to 30 nodes).

In all networks RTD exhibited sporadic outliers exceeding average value up to more than 20 times. Except from 3G-VPN environments we suspect that these outliers were caused by temporary overload of CPU (100 % load). Number of such outliers was significantly higher when main processor was running at 600 MHz compared to 800 MHz clock frequency.

OPC UA server is a computational performance demanding application. In case full speed interrogation by OPC UA client is executed, 800 MHz ARM Cortex A8 processor was only enough to handle OPC UA server execution without affecting round-trip delay in data transfer channel.

REFERENCES

[1] Z. Qian, W. Ruicong, Ch. Qi, "IOT Gateway: bridging wireless sensor networks into Internet of Things", in *Proc. IEEE/IFIP 8th Int. Conf. Embedded and Ubiquitous Computing (EUC)*, 2010, pp. 347–352.

[2] X. Yong, W. Chen, "Design and implementation of gateway for vehicular networks", in *Proc. 25th Chinese Control and Decision Conf. (CCDC)*, 2013, pp. 4486–4489.

[3] Ji-De Huang, Han-Chuan Hsieh, "Design of gateway for monitoring system in IoT networks", in *Proc. IEEE Int. Conf. and IEEE Cyber. Physical and Social Computing,* 2013, pp. 1876–1880. [Online]. Available: https://doi.org/10.1109/greencom-ithings-cpscom.2013.348

[4] S. Guoqiang, C. Yanming, Z. Chao, "Design and implementation of a smart IoT gateway", in *Proc. IEEE Int. Conf. and IEEE Cyber. Physical and Social Computing*, 2013, pp. 720–723. [Online]. Available: https://doi.org/10.1109/greencom-ithings-cpscom.2013.130

[5] M. Jung, J. Weidinger, C. Reinisch, "A transparent IPv6 multi-protocol gateway to integrate building automation systems in the Internet of Things", in *Proc. IEEE Int. Conf. Green Computing and Communications (GreenCom)*, 2012, pp. 225–233. [Online]. Available: https://doi.org/10.1109/greencom.2012.42

[6] Z. Shelby, "Embedded web services", *IEEE Wireless Communications*, vol. 17, no. 6, pp. 52–57, 2010. [Online]. *Available:* https://doi.org/10.1109/MWC.2010.5675778

[7] "OPC Unified Architecture: Interoperability for Industrie 4.0 and the Internet of Things". OPC Foundation Brochure. [Online]. Available: http://www.opcfoundation.org.

[8] L. Yuangua, X. Haixia, Y. Wei, Z. Xingwu, "Design of Zigbee gateway in intelligent monitoring system for agriculture", in *Proc. Int. Conf. Electric Engineering and Computer Mechatronic Science (MEC),* 2011, pp. 2213–2216.

[9] Y. Fuxing, Y. Chuansheng, "Design of WSN gateway based on ZigBee and TD", in *Proc. Int. Conf. Electronics and Information Engineering (ICEIE)*, 2010, pp. 76–80.

[10] J. Folkens, "Building a gateway to the Internet of Things, Texas instruments", Texas instruments White paper 2014. [Online]. Available: http://www.ti.com/lit/wp/spmy013/spmy013.pdf

[11] Intelligent Gateways Play a Key Role in the Internet of Things (IoT). Intel Solution Brief, 2014. [Online]. Available: http://www.intel.com

[12] S. Cavalieri, G. Cutuli, "Performance evaluation of OPC UA", in *Proc. IEEE Conf. Emerging Technologies and Factory Automation (ETFA 2010)*, 2010, pp. 1–8. [Online]. Available: https://doi.org/10.1109/etfa.2010.5641184

[13] S. Cavalieri, F. Chiacchio, "Analysis of OPC UA performances", *Computer Standards & Interfaces*, vol. 36, no. 1, pp. 165–177, 2013. [Online]. Available: https://doi.org/10.1016/j.csi.2013.06.004

[14] C. Seung-Seok, C. Sung-Hyuk, C. T. Charles, "A Survey of binary similarity and distance measures", *Journal of Systemics, Cybernetics and Informatics*, vol. 8, no. 1, pp. 43–48, 2010.