

Applicability of OPC to real-time data collection: simulation study

Ari Koponen,* Manu Huttunen, Riku Pöllänen and Olli Pyrhönen
Department of Electrical Engineering
Lappeenranta University of Technology
P.O. Box 20, 53851 Lappeenranta, Finland

Abstract

OPC (OLE for Process Control) provides an easy way to connect simulation systems to control systems, but there has been little research on the reliability and timing accuracy of OPC servers.

Actual sampling time of data can be determined using a fast counter as a variable. Variations in the sampling time can be examined using consecutive sampled counter values to construct jitter values. Difference in sampling time between variables can be observed by reading the same counter value from different server channels.

In this study the timing accuracy of one specific OPC server is considered and reliability of variable values assessed. Accuracy of the time stamp provided by the OPC server is evaluated and network and CPU load monitored. Tests are conducted using different settings at both server and client side.

According to the results the timing accuracy and variable reliability varies depending on the server and client settings. By using correct settings the timing accuracy of the server is good and the server provides correct values. Settings also affect the accuracy of the time stamp as well as the CPU and network load.

Keywords: Data collection, OPC, timing accuracy, reliability.

1 Introduction

Software has an essential role in automation and control systems in field and process management levels of the process industry as well as in business management level. The automation and control software is run on various field devices, process stations and PLCs provided by different vendors. This software needs to be accessed i.e. from SCADA applications and the

business management level of the plant. The diverseness of the automation and control equipment creates a need for an easy and efficient way of connecting to these devices.

The OPC specifications that describe standard communication mechanisms for devices and software in a process control system are meant to meet the above mentioned requirements. The OPC interface has gained extensive ground in the industry and many manufacturers of DCS systems, PLCs and controllers provide an OPC interface to their systems. It has gained the status of a worldwide industrial de facto standard [1].

The OPC interface can also be used for connecting simulation systems to automation and control systems if both systems support the same OPC specification and if their versions are compatible. This kind of connection can be used for operator training and automation design and testing purposes [2, 3].

The OPC architecture is a client-server model where the server is responsible for data collection from the physical device. The client application uses the OPC interfaces specified by the OPC Foundation [4] to communicate with the server.

The performance of an OPC server has been tested e.g. in [5], where the throughput of an OPC server was considered. To our knowledge properties of an OPC server such as variation in sampling time and timing accuracy have not been studied as widely as throughput. These properties are important to real-time applications such as real-time simulation environments.

Variation in sampling time can be expressed by calculating jitter values from consecutive sampled values of a fast counter. Jitter J is calculated as

$$J(k) = I(k) - I(k-1) - T_s \quad (1)$$

where I is the counter value, k is the sample index and T_s is the sampling time.

*Corresponding author. E-mail: akoponen@lut.fi

2 OPC Data Access Specification

The current version of the OPC Data Access Specification is 3.0, but the client software used in this simulation study supports mainly the OPC Data Access Specification 1.0A. The OPC Data Access Specification 1.0A specifies two data collection ways for the OPC server [6]. In *device* mode the server reads the value of a requested item directly from the device. In this mode the reading can be expected to be slow and is mainly intended for diagnostics and very critical applications. In *cache* mode the server reads the values of specified items to its cache using a sampling time obtained from the client.

An OPC client can access data items by using one of the three alternative reading modes offered by the OPC server. These modes are *asynchronous*, *synchronous* and *callback*. In the asynchronous mode the client sends a request for a server but does not wait for a response. When using the synchronous mode the reading operation is conducted in a single transaction and the client waits for a response before continuing its task. In the callback mode a client can subscribe to items accessible through the server and instruct it to update the values of these items using a sampling time provided by the client. A callback¹ is sent to the client only when the change in value exceeds the deadband criteria assigned for that item. After receiving a callback the client can read all the requested items from the server.

3 Description of test arrangement

3.1 Hardware and software

The simulation was conducted using a test arrangement illustrated in Fig. 1. A fast counter that increments its value in intervals of one millisecond was implemented to the PLC. The OPC server reads the value of the counter to multiple channels, the number of which was altered in different simulations. Using an OPC connection the value of the counter was read from all the channels using two different sampling times and stored with Matlab. The PLC used in this simulation was Beckhoff TwinCAT v2.9.0 soft-PLC that runs on a commercial desktop PC together with Windows 2000 operating system. The OPC server was TwinCAT OPC Server v4.1.0.37 supporting the Data Access 2.0. The used Matlab version was 6.5 and the

OPC client was IPCOS's OPC for Matlab, DLL version 3.0.0.19.

The IPCOS's OPC client supports OPC Data Access Specification 1.0A with the exception of asynchronous reading mode. In addition to the properties specified in the specification it can read several channels at once to its own buffer (client cache), which decreases network load due to smaller communication overhead and decreases the difference in time between the values of different channels. The OPC client also enables better control of the sampling time compared to Matlab's standard functions.

3.2 Test simulations

The purpose of the simulations was to examine the effects of different settings and to find the best settings for both server and client side to minimize the jitter and the network and CPU load. A data collection application based on Matlab and IPCOS's OPC client, and capable of conducting the simulations with different settings was developed. The complete list of the simulations and settings is presented in Table 1. The simulation procedure was designed to address all the 48 combinations of the following settings: supported client reading modes (synchronous/callback), server mode (device/ cache), client cache on/off, sampling time (0.5 s or 2 s), number of channels (5, 20 or 200). In each simulation 1000 counter values per channel, the calculated jitter values, the time stamps provided by the OPC server and the information about the system load were recorded by the client program. From this data the maximum difference in time between the data read at the same client sample time from different channels was calculated for each simulation. Also the accuracy of the time stamp provided by the OPC

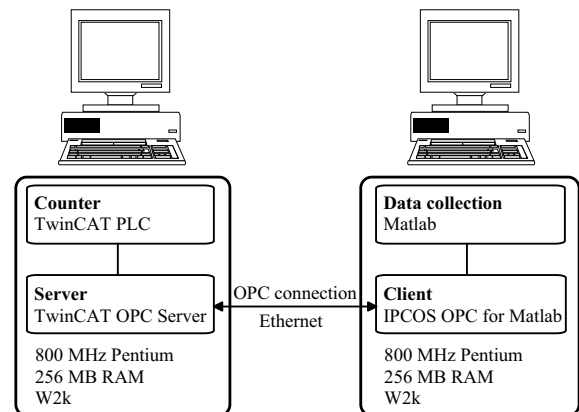


Figure 1: A schematic of the simulation arrangement.

¹An indication of a changed value.

Table 1: Server and client settings for the test simulations. Reading modes (RM) s and c stand for synchronous and callback reading mode respectively. Server modes (SM) c and d stand for cache and device respectively. CC stands for client cache.

No	T_s [ms]	Samples	RM	SM	CC	Channels
1	500	1000	s	c	on	5
2	500	1000	s	c	on	20
3	500	1000	s	c	on	200
4	500	1000	s	c	off	5
5	500	1000	s	c	off	20
6	500	1000	s	c	off	200
7	500	1000	s	d	on	5
8	500	1000	s	d	on	20
9	500	1000	s	d	on	200
10	500	1000	s	d	off	5
11	500	1000	s	d	off	20
12	500	1000	s	d	off	200
13	500	1000	c	c	on	5
14	500	1000	c	c	on	20
15	500	1000	c	c	on	200
16	500	1000	c	c	off	5
17	500	1000	c	c	off	20
18	500	1000	c	c	off	200
19	500	1000	c	d	on	5
20	500	1000	c	d	on	20
21	500	1000	c	d	on	200
22	500	1000	c	d	off	5
23	500	1000	c	d	off	20
24	500	1000	c	d	off	200
25	2000	1000	s	c	on	5
26	2000	1000	s	c	on	20
27	2000	1000	s	c	on	200
28	2000	1000	s	c	off	5
29	2000	1000	s	c	off	20
30	2000	1000	s	c	off	200
31	2000	1000	s	d	on	5
32	2000	1000	s	d	on	20
33	2000	1000	s	d	on	200
34	2000	1000	s	d	off	5
35	2000	1000	s	d	off	20
36	2000	1000	s	d	off	200
37	2000	1000	c	c	on	5
38	2000	1000	c	c	on	20
39	2000	1000	c	c	on	200
40	2000	1000	c	c	off	5
41	2000	1000	c	c	off	20
42	2000	1000	c	c	off	200
43	2000	1000	c	d	on	5
44	2000	1000	c	d	on	20
45	2000	1000	c	d	on	200
46	2000	1000	c	d	off	5
47	2000	1000	c	d	off	20
48	2000	1000	c	d	off	200

server was evaluated.

4 Results

Histograms of the jitter values of the first channel for simulations 7-12 and 25-48 are presented in Fig. 2-6. The values above the histograms are the simulation number (referring to Table 1), sampling time in seconds, reading mode, server mode, client cache mode, number of channels, minimum and maximum jitter value and maximum difference in time between channels. The last value indicates how close the sampling times of the channels are to each other.

The simulations were first examined by maximum jitter values and maximum difference between channels. The simulations where the maximum jitter value was smaller than 500 milliseconds and the maximum difference between channels smaller than 10 milliseconds were simulations 8, 9, 19-21, 31-33 and 43. In these simulations the server was in device mode and client cache was turned on. In simulations 7-9, 19-21 and 31-33 the distributions looked fairly good and the difference in sampling times between channels and the maximum jitter values were small for all tested number of channels.

The reading mode's effect on jitter values was examined by setting the server to device mode and client cache on. Fig. 7 illustrates the mean, standard deviation and median values of jitter with sampling time of 500 milliseconds at the upper row and 2000 milliseconds at the lower row. The mean values are a few milliseconds for both reading modes regardless of the number of channels. Differences in the standard deviation values can be seen between the reading modes.

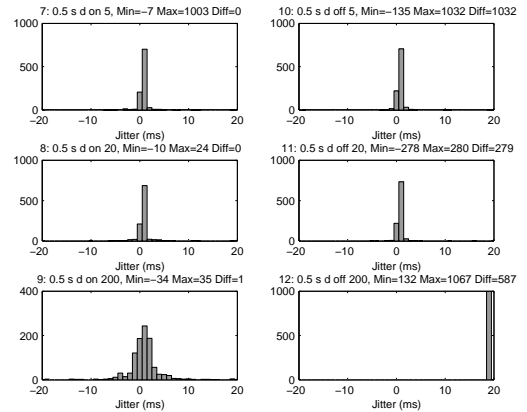


Figure 2: Jitter histograms for simulations 7-12.

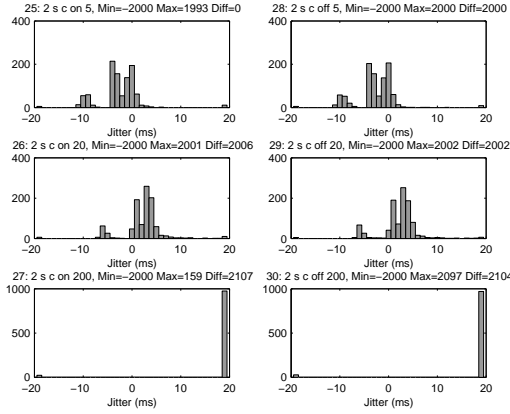


Figure 3: Jitter histograms for simulations 25-30.

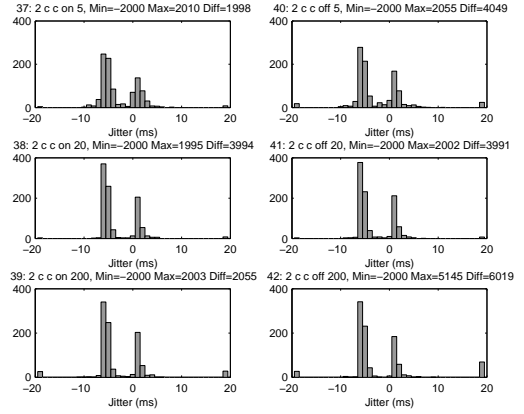


Figure 5: Jitter histograms for simulations 37-42.

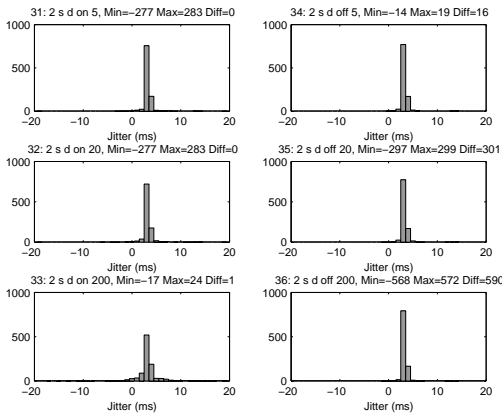


Figure 4: Jitter histograms for simulations 31-36.

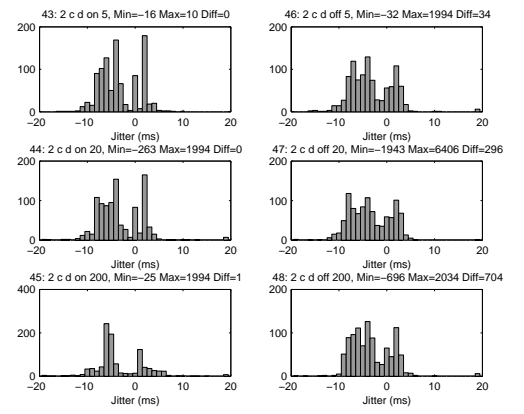


Figure 6: Jitter histograms for simulations 43-48.

The large standard deviation of the synchronous reading mode with five channels can be explained by an outlier of 1003 milliseconds that appears in all five channels. If this value is excluded the standard deviation is 2 milliseconds. With the 2000 millisecond sampling time the standard deviation of the callback reading mode is significantly larger than the standard deviation of the synchronous reading mode. The same occurrence can be seen from simulations 31-33 and 43-45.

In simulation 12 all the values were late for over a sampling time. The reason for this is that the data transfer took over 500 milliseconds. Because of the synchronous reading mode the OPC client waited for the next full sampling time instead of starting the data transfer immediately.

At first glance the simulations 27 and 30 seem to behave similarly as the simulation 12. However, this is not the case. Namely, Fig. 3 is a bit misleading because the mean value of the jitters in simulations 27

and 30 were actually about 50 ms and hence the histograms show that almost all jitters were as least 20 ms. The axis settings for these plots were however chosen to be the same as for the rest of the figures for the sake of consistency.

What is also notable is that whenever the server cache is turned on the minimum jitter value equals to $-T_s$. This means that the first two terms in Eq. 1 have to be equal and hence the consecutive counter samples are same. A closer look to the data revealed that occasionally there appears to be sampling events when most of the channels were not updated (results $-T_s$ as jitter). This indicates that there might be either synchronization problems between client and server in this mode or a some kind of server design flaw.

The amount of received and sent data for the client's Ethernet card is presented in Fig. 8. In simulations 3, 9, 15 and 21 the network load is considerably lower than in the other simulations. The common factor for the simulations with lower network load is the use of

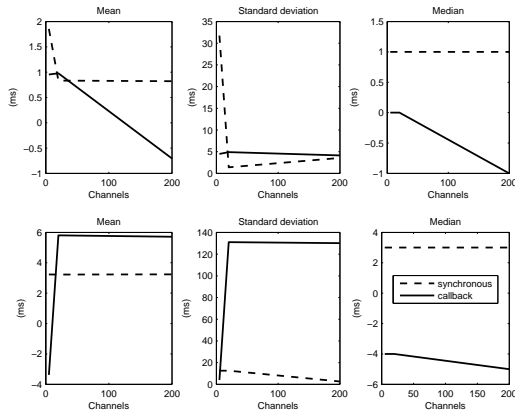


Figure 7: Mean, standard deviation and median values for jitter.

client cache.

The server's CPU load was examined for all the simulations (see Fig. 9). In simulations 5, 6, 17 and 18 (20 and 200 channels) the CPU load is above average. In these simulations the server mode is set to cache and the client cache off. When the client cache is turned on (simulations 2, 3, 14 and 15) the CPU load is halved. If the server operates in device mode the client cache setting does not have such a notable effect. In the figure for 5 channels the elevated CPU load after simulation 25 could be caused by statistical anomalies etc., because the same phenomenon does not appear with 20 or 200 channels.

The reliability of the time stamp provided by the OPC server was assessed by calculating jitter values from the time stamp data (see Fig. 10). The calculated maximum and minimum values for jitter and the maximum difference between channels correspond fairly well to the values calculated from the counter data. The jitter values calculated from the time stamps are not as evenly distributed as the values calculated from the counter values. This is because the time resolution of the time stamp is 10 milliseconds.

5 Conclusions

The best performance of the OPC software used in this study was achieved when the server was operated in device mode and synchronous reading mode and cache was used at the client side. Using the settings shown in Table 2 the jitter as well as the network and server CPU load are the smallest.

The client cache, however, is a feature that is not spec-

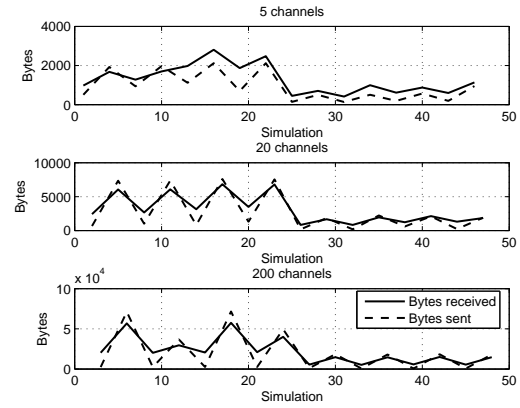


Figure 8: Network loads for the simulations.

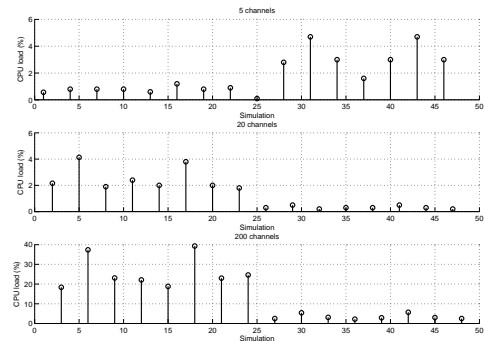
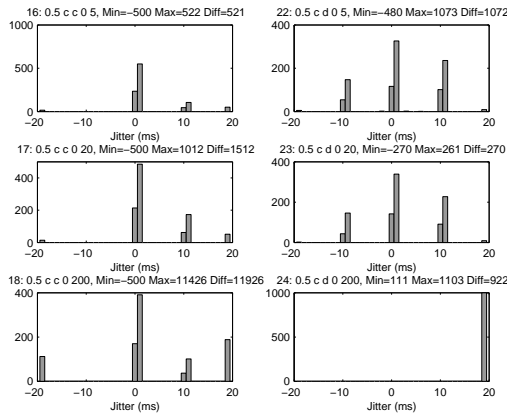


Figure 9: CPU loads for the simulations.

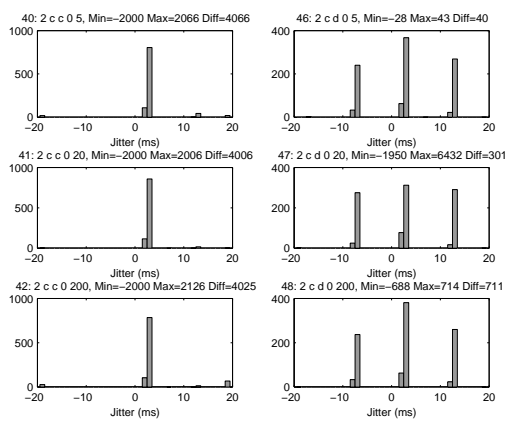
ified in the Data Access Specification. By using client cache the network and server CPU load is reduced and the difference between values read from different channels is reduced close to zero in most of the simulations. When client cache was turned off and synchronous reading mode used, better sampling time accuracy was achieved by using the server in device mode at least with 2000 millisecond sampling time. If using the callback reading mode without client cache the sampling time was closer to the desired value when the server was run in device mode. Without client cache the best distribution for the jitter and lower server CPU load was achieved by using synchronous reading mode and device mode for the server although in some cases

Table 2: The best settings for the tested OPC software.

Reading mode	Server mode	Client cache
synchronous	device	on



(a)



(b)

Figure 10: Histograms for jitter values calculated from the time stamp data of the simulations. (a) simulations 16-18 and 22-24, (b) simulations 40-42 and 46-48.

the maximum difference between channels might be larger with these settings than others.

Only one specific OPC server was used in this study and the results are likely to vary to some extent with different servers depending on the implementation. In order to draw more general conclusions more server/client combinations should be tested. However,

at least with this specific pair of OPC server and client the reliable and accurate performance for real-time data collection was achieved.

References

- [1] R. Pattle and J. Ramisch, "OPC the de facto standard for real time communication," in *Proceedings of the Joint Workshop on Parallel and Distributed Real-Time Systems*, 1-3 April 1997, pp. 289–294.
- [2] J. Lappalainen, S. Tuuri, T. Karhela, J. Hankimäki, P. Tervola, S. Peltonen, T. Leinonen, E. Karppanen, J. Rinne, and K. Juslin, "Direct connection of simulator and DCS enhances testing and operator training," in *Proceedings of TAPPI 1999 Engineering / Process & Product Quality Conference*, Hilton Anaheim, Anaheim CA, Sept. 12–16, 1999, pp. 495–502.
- [3] J. Rinta-Valkama, M. Välisuo, T. Karhela, P. Laakso, and M. Paljakka, "Simulation aided process automation testing," in *IFAC's Conference on Computer Aided Control System Design (CACSD)*, University of Salford, UK, Sept. 11–13, 2000.
- [4] (2005) The OPC foundation. [Online]. Available: <http://www.opcfoundation.com/>
- [5] J. Peltoniemi, T. Karhela, and M. Paljakka, "Performance evaluation of OPC-based I/O of a dynamic process simulator," in *The Proceedings of the 2001 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*. SCS, 15-19 July 2001, pp. 231–236.
- [6] OPC Foundation, *OLE for Process Control Data Access Standard (UPDATED)*, 1.0A ed., Sept. 1997.