Industrial Internet of Things for Resource Constraint Devices

Viraj Kulkarni
Technical University of Darmstadt
Email: viraj.kulkarni@stud.tu-darmstadt.de

David Abuladze
Technical University of Darmstadt
Email: david.abuladze@stud.tu-darmstadt.de

***Abstract*—The abstract goes here.**

I. Introduction

This demo file is intended to serve as a "starter file" for IEEE conference papers produced under LaTeX using IEEEtran.cls version 1.8b and later. I wish you the best of success.

mds

August 26, 2015

A. Reducing the network footprint

Dürkop et al (2012) have identified that the memory footprint for current OPC UA was too high for small devices [2]. In this work [2] two Service Oriented Architectures (SOA) for autoconfiguration of real-time ethernet systems have been introduced. The first Architecture is Device Profile for Web Service (DPWS) and the second is OPC UA. In addition to these SOAs, memory footprint for OPC UA implementation has been discussed. It has been emphasized, that significantly less program and data memory was needed for OPC UA, and number of code lines was less than a half of the DPWS. For both DPWS and OPC UA servers the Tiger Profinet Solution (TPS-1) has been chosen. It was extended the firmware by OPC UA stack using binary encoding. The OPC UA server was implemented on the Nano Embedded Device Server profiles. The binary transport profile was assigned, and the authentication using certificates have been used for the support. Dürkop et al (2012) have concluded that OPC UA requires significantly less hardware resources. Authors have emphasized that OPC UA server was developed from scratch in order to achieve small memory footprint for embedded devices. Further, in the article [8] it has been discussed how to minimize the OPC UA network footprint in order to fit Bluetooth Low energy (BLE) frame. In paper [5] authors have implemented TCP-1 with real-time Ethernet interfaces, but the problem with devices with different transport protocol technologies, such as BLE, CAN bus, has appeared. The cause of this problem is low payload size of these protocol technologies. That is the reason why the authors [8] have suggested to optimize OPC UA information model to suite it for BLE and CAN bus. In their article, authors have proposed to optimize the size of OPC UA in ReadResponse fields. OPC UA Transport Profile (OOTP) with TCP/IP protocol has been implemented. As a result, OPC UA optimized header uses only 7 bytes and there are 7 bytes left for payload. This article has presented the possibility to integrate the OPC UA network footprint to the different small devices.

UA Fields	**Actual Size (byte)**	**Proposed Optimized Size**	**Remarks**
Message Type	3	3 bit	6 type of message and can be uniquely represented by 3 bit
Chunk Type	1	Excluded	Handled by Gateway
Message Size	4	5 bit	Message size is always less than 31 bytes
SecureChannelId	4	Excluded	Handled by Gateway
Security Token Id	4	Excluded	Handled by Gateway
Security Sequence Number	4	Excluded	Handled by Gateway
Security RequestId	4	Excluded	Handled by Gateway
Message : Encodeable object			
TypeId : ExpandedNodeId			
Encoding Byte	1	Excluded	Always Numeric
Namespace	1	1 byte	No change
Identifier	2	1 byte	Supports only up to 255 nodes and services
ReadResponse			
ResponseHeader : ResponseHeader			
Timestamp	8	Excluded	Will be added in the gateway. Difference of just 6ms
RequestHandle	4	2 bit	Reserved for multiple reads
ServiceResult	4	1 bit	Only says if the result is good or bad
ServiceDiagnostics : DiagnosticInfo			
EncodingMask	1	1 bit	Tells if the diagnostic info is available or not
StringTable[]	4	4 bit	Vendor specific diagnostic information
AdditionalHeader : ExtensionObject			
TypeId : ExpandedNodeId			
Encoding	1	Excluded	Not used. Reserved for future use.
Identifier	1	Excluded	Not used. Reserved for future use.
EncodingMask	1	Excluded	Not used. Reserved for future use.
Total	**52 bytes**	**4 bytes**	
ArraySize	4	Excluded	Array size will be fixed to 2 (NodeId and the actual value).
[0]:DateValue . [15]:DataValue	Depend on read data	Max. data size of 10 bytes	The data headers take 3 bytes leaving maximum payload size of 7 bytes. Please refer to Table 3 for details.
Array of DiagnosticInfos			
ArraySize	4	Excluded	Handled by Gateway

Figure 1. Message size dependence on the used encoding from [9], p.1574

B. *Efficient OPC UA encoding*

The next improvement area for embedded devices is a possibility to encode data. It allows transferring data for devices with limited access memory [4]. Reduced bandwidth becomes a very important aspect, because it is increasing use of "smart" field devices in industrial networks and there is also semantic integration for device layers [4]. Article [9] integrate different performance optimization techniques into the OPC UA. This include asynchronous communication and different encoding possibilities). OPC UA has been working in the IEC 61850-OPC UA mapping system (IEC 61850 is communication protocol which is international defined). Proposed possibilities have been analyzed in simulation environment. OPC UA communication is based on XML-based web service. To reduce message size two encoding technologies where discussed, such as EXI and Binary. There are several articles which highlight this technologies [4] [9].First [9] article discuses both XML Interchange (EXI) and Binary encoding and second only binary encoding. The EXI is a design for high performance parsing, which contains main benefits of both the XML and UA Binary - the binary encoding is the most efficient and it has less message size, but EXI is web technology and have good tool support [9].

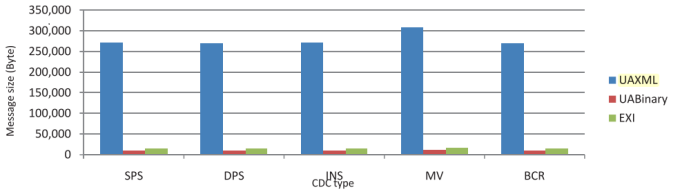

Figure 2. Message size dependence on the used encoding from [9], p.1574

In Figure 1 shows, that UA binary and EXI encoding have significant reduced message size compared to the XML. But between EXI and Binary is not much difference. The next article [4] describes the binary encoding for limited memory devices. Main focus was on single-chip microcomputing platforms. Goal was to reduce bandwidth to (116 kB for Namespace 0). It was clarified that, if we want to improve bandwidth, it is possible only if data is compressed as a service[4]. Parsing XML data are problematic and it is necesary to parse data in memory to access it efficiently. That was the reason why authors presented OPC UA hardware server IP Core to overcome the limitation. To use this technology binary encoding was created. As a result binary encoding was suitable for small devices such as 8 bit Electrically Erasable Programmable Read-Only Memory (EEPROM) and Flash memory products. String compression demonstrated that the memory requirements have reduced from 197 kb to 116 kb, which corresponds to the 56% of improvement[4].

C. *Efficient address space*

This problem is addressed in the paper "Efficient address space generation for an OPC UA Server" by Girbea et al (2011) [1]. Address space improvement is a very important aspect to get the speedup for development and maintenance time. In order to understand the structure of OPC UA Serve, the startup procedure has been discussed. OPC UA Server startup consists of multiple steps, as presented on the Figure 3.

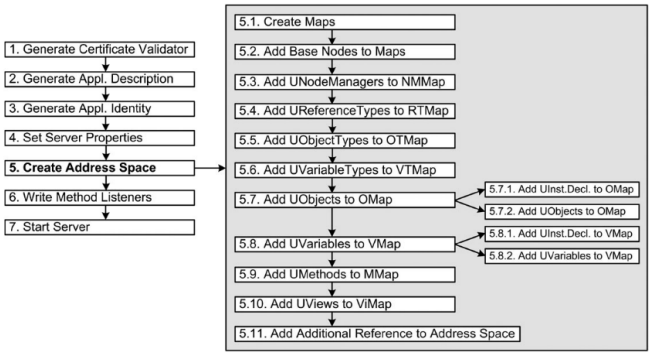

Figure 3. Wide single column figure in a twocolumn document.

On the left-hand side of the Figure 3 there are main steps to be programmed. The list of the actions for the address space generation are presented on the right-hand side of the Figure 3 [1]. The authors claim that the main units of address space are nodes and connected references. The base nodes are abstract and all the other nodes are subparts of base nodes, such as variable, variabletype, object objecttype, data view and method. There are 4 algorithms have been implemented, created for type nodes, for methods, for view and for object. There can be some similarities in these 4 algorithms. Nevertheless, the details description of each algorithm is not relevant for this seminar paper. The first algorithm uses referencetype hierarchy and have several similar properties with objecttypes and variabletypes. The second algorithm uses object nodes hierarchy and is similar to the variable nodes. The third algorithm is based on method nodes. This algorithm adds the nodes in success groups. The fourth algorithm generates the view nodes hierarchy. Algorithm 5 is simple and it reads the source, gets target node and reference and after that adds reference to address space[3].

Test scenario	Number of nodes	Time without algorithms	Normalized time without algorithms	Time with algorithms	Normalized time with algorithms	Improvement (speedup)
Air conditioner	70	365	521	157	224	2.32
Conveyor system	100	495	495	217	217	2.28
Station 4	200	958	479	430	215	2.23

Figure 4. Improvement of development time from [3] p. 555.

Test scenario	Time without algorithms	Time with algorithms	Improvement (speedup)
Air conditioner	34	21	1.62
Conveyor system	32	22	1.45
Station 4	35	24	1.46

Figure 5. Improvement of maintenance time from [3] p. 556.

Figure 3 shows that the main advantage of the presented algorithms is reduced development time. Speedup was around 2.2-2.3

Figure 4 shows that maintenance time speedup was 1.4-1.6

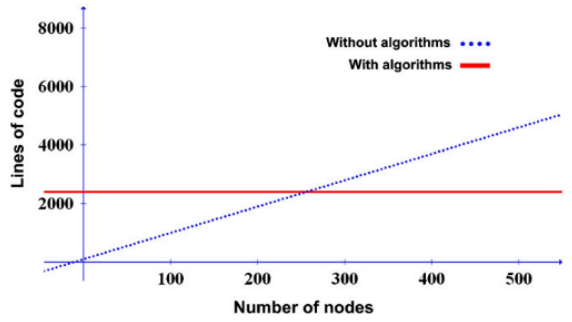

Figure 6. From [3] p. 555 Lines of code used to generate the address space.

Figurem 8 shows that line of code is constant and independent of node number. This means that the more nodes the more efficient will be the algorithm.

results of implemented algorithms shows significant improvement in comparison to the address space generaion without the algorithms, presented above. The nodes can also be modified by UA Server Admin in online or offline mode and by client in online mode.

D. Security

One of the main parameters of the OPC UA specification that has a significant impact on performance of the resource constrained devices is security. As stated in the part 7 ———cite OPC UA specifications part 7——- of the OPC UA specifications, OPC UA provides three security profiles: Basic128Rsa15, Basic256 and None. Since OPC UA is used at different levels of automation in disparate applications within the same environment, a good trade off between the security and performance should be reached. This can bne done by taking into consideration, the level at which OPC UA is implemented. For example, At the very high level, where the device resources are abundant and where security might be more important than the performance, the profile which provides the highest amount of security should be used. Where as at very low levels, like in embedded devices where the device resources are very scarce, but security might not be as important as the performance, a weaker security profile which favors performance should be used.

Based on this premise, Olli Post, Jari Seppälä and Hannu Koivisto have measured the performance of resource constrained devices in the paper ”The performance of OPC UA security model at field device level”. The findings of the evaluation are as follows. [7]

For the evaluation of the performance in terms of efficient data transfer, all the three security profiles of the OPC UA specification namely None, Basic256 and Basic128Rsa15 as well as IPSec AH were compared. The evaluation was done to asses the delay caused by security measures with three packet sizes 1024 bytes, 10 240 bytes and 10 400 bytes.

IPSec is a network layer protocol that guarantees data confidentiality and integrity, origin identification and prevent replay attacks (Douligeriset al., 2007)————————cite this ref properly————————-

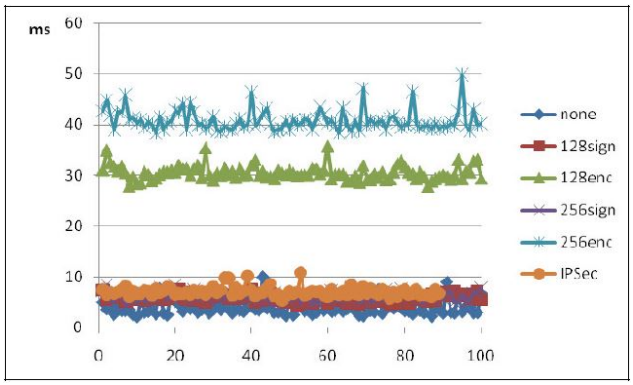

Figure 7. Comparison between different OPC UA security profiles with 102400 bytes

The first evaluation was done on 10 240 bytes which is represented in the figure 2 ———— insert the figure and cite it properly————————-. The figure shows the delay caused by none, 128signed, 128encrypted, 256signed, 256encrypted and IPSec profiles. It can be seen in the figure that there is a tremendous delay when the data was encrypted in cases of 128encrypted and 265encrypted as opposed to the security profile None. Hence based on the evaluation, it is safe to say that in field or embedded devices which are strapped for resources and where confidentiality of the data is not a strict requirement, encryption adds a significant delay and is not an optimal solution for efficient data transfer.

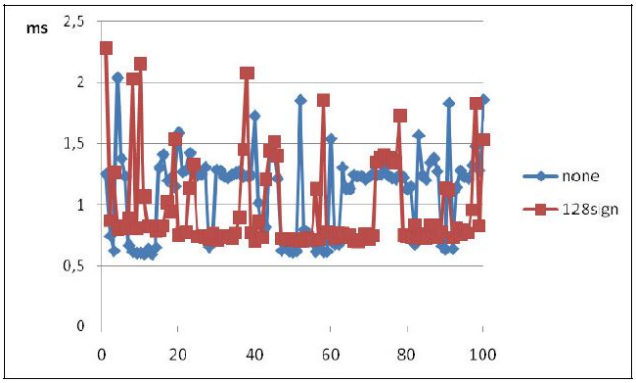

Figure 8. Delay caused in a data size of 1024 bytes

The next performance measurements were done with a small data size of 1024 Bytes. The figures 3 and 4 show the delay caused in sending packets with 1024 bytes. The figure 3 and 4 depict the comparison between the profiles None, Basic128 and the Basic256 , IPSec AH. It can be inferred from the diagrams that authentication does not have a big impact in terms of delay in small packets of 1024 bytes.

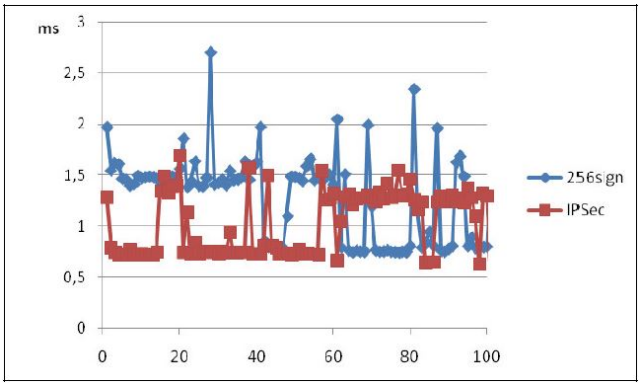

Figure 9. Delay caused in a data size of 1024 bytes

1) Subsubsection Heading Here: Subsubsection text here.

E. Scalability

OPC UA is used to enable Internet of Things for industrial automation applications which provides powerful information representation that is exchangeable through inter operable services. The scalability of OPC UA refers to the ability of the hardware devices at different levels in the same environment to implement OPC UA efficiently. The problem with most of the current implementations of OPC UA is that they come with a lot of features which might not be needed by a hardware device at a certain level. Since each hardware device does not have the capability to implement every feature of OPC UA, the implementation of all the features including the ones which are not used by the device puts unnecessary strain on the device in terms of memory, network overhead etc. Some off the shelf implementations of OPC UA take as much as 10MB of memory. [6] The memory required by the OPC UA implementation is directly proportional to the number of features implemented and the complexity of those features. But some devices in the environment may have as less as a few kilo bytes of memory. Hence off the shelf implementation of OPC UA is not suitable for such devices. New stripped down implementations need to be tested for such resource limited devices.

The paper ”Scalability of OPC-UA down to the chip level enables Internet of Things” by Jahanzaib Imtiaz and Jürgen Jasperneite investigates one such implementation of OPC UA for resource limited devices based on the ”Nano Embedded Device Server Profiles” of the OPC foundation.

II. Conclusion

The conclusion goes here.

Acknowledgment

The authors would like to thank...

References

[1] S. Cavalieri and G. Cutuli. Performance evaluation of opc ua. In *2010 IEEE 15th Conference on Emerging Technologies Factory Automation (ETFA 2010)*, pages 1–8, Sep. 2010.

[2] L. Dürkop, J. Imtiaz, H. Trsek, and J. Jasperneite. Service-oriented architecture for the autoconfiguration of real-time ethernet systems. In *3rd Annual Colloquium Communication in Automation (KommA)*, 2012.

[3] A. Girbea, S. Nechifor, F. Sisak, and L. Perniu. Efficient address space generation for an opc ua server. *Software: Practice and Experience*, 42(5):543–557, 2012.

[4] C. P. Iatrou and L. Urbas. Efficient opc ua binary encoding considerations for embedded devices. In *Industrial Informatics (INDIN), 2016 IEEE 14th International Conference on*, pages 1148–1153. IEEE, 2016.

[5] J. Imtiaz and J. Jasperneite. Scalability of opc-ua down to the chip level enables “internet of things”. In *Industrial Informatics (INDIN), 2013 11th IEEE International Conference on*, pages 500–505. IEEE, 2013.

[6] J. Imtiaz and J. Jasperneite. Scalability of opc-ua down to the chip level enables “internet of things”. In *2013 11th IEEE International Conference on Industrial Informatics (INDIN)*, pages 500–505, July 2013.

[7] O. Post, J. Seppälä, and H. Koivisto. The performance of opc-ua security model at field device level. In *ICINCO-RA*, pages 337–341, 2009.

[8] G. M. Shrestha, J. Imtiaz, and J. Jasperneite. An optimized opc ua transport profile to bringing bluetooth low energy device into ip networks. In *Emerging Technologies & Factory Automation (ETFA), 2013 IEEE 18th Conference on*, pages 1–5. IEEE, 2013.

[9] S. Sucic. Optimizing opc ua middleware performance for energy automation applications. In *Energy Conference (ENERGYCON), 2014 IEEE International*, pages 1570–1575. IEEE, 2014.