Computer Standards & Interfaces 36 (2013) 165–177

Contents lists available at ScienceDirect

Computer Standards & Interfaces

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

Analysis of OPC UA performances

Salvatore Cavalieri *, Ferdinando Chiacchio

University of Catania, Department of Electrical Electronic and Computer Engineering, Viale A.Doria, 6, 95125 Catania, Italy

ARTICLE INFO

Article history:
Received 18 December 2010
Received in revised form 14 December 2012
Accepted 28 June 2013
Available online 16 July 2013

Keywords:
OPC
OPC UA
Performance evaluation

ABSTRACT

OPC UA is the evolution of the well known OPC COM and XML specifications. OPC UA adopts a very complex software infrastructure to realise the communication among industrial applications; furthermore it features many mechanisms realising data exchanges, whose tuning depends on several parameters. The aim of this paper is to deal with the performance evaluation of OPC UA. The main data exchange mechanisms which may influence performance of the client/server communications will be pointed out; then, the analysis of the overhead they introduce will be presented and discussed. Finally, some guidelines about the setting of OPC UA mechanisms will be given on the basis of the results achieved.

1. Introduction

OPC Unified Architecture (OPC UA) is the current OPC Foundation's technology for secure, reliable and interoperable transport of raw data and pre-processed information from the shop floor into production planning systems [1].

Definition of OPC specifications started more than fifteen years ago to simplify and to standardise data exchange between software applications in industrial environment. The rapid diffusion of the first version of OPC specifications was due to the choice of Microsoft's DCOM as the technological basis. However, exactly this point raised the majority of criticism regarding OPC; OPC technology was too focused on Microsoft, platform-dependent and not firewall-capable, and thus not suitable for use in cross-domain scenarios and for the Internet. When XML and Web Services technologies have become available, the OPC Foundation adopted them as an opportunity to eliminate the shortcomings of DCOM. Since 2003 the OPC XML Data Access (DA) specification has offered a first service-oriented architectural approach besides the "classic" DCOM-based OPC technology; this Web services-based concept enabled applications to communicate independently of the manufacturer and platform.

Few years ago, the OPC Foundation has introduced the OPC UA standard which is based on a service-oriented, technology- and platform-independent approach, creating new and easy possibilities of communicating with Linux/Unix systems or embedded controls on other platforms and for implementing OPC connections over the Internet. The new possibilities of using OPC components on non-Windows platforms, embedding them in devices or implementing a standardised OPC communication across firewall boundaries allow speaking of a change of paradigms in OPC technology. OPC UA servers can be varied and scaled in their scope of functions, size, performance and the platforms they support. For embedded systems with limited memory capacities, slim OPC UA servers with a small set of UA services can be implemented; at the company level, in contrast, where memory resources are not that important, very powerful OPC UA servers can be used with the full functionality. OPC UA specifications now offer a security model, which wasn't available in the previous versions of OPC specifications; the OPC UA security governs the authentication of clients and servers and ensures data integrity, trustworthiness and authorisation within OPC communication relationships.

All the features offered by OPC UA specifications are realised by introducing a complex software infrastructure made up by several layers each offering a particular set of functionalities; the organisation based on layers has the main advantage to allow interchangeability between their implementations, assuring technology-independence. Each of the layers of the OPC UA communication stack features many mechanisms realising data exchanges, whose tuning depends on several parameters. Both the intrinsic complexity of the stack implementation of OPC UA and the difficulties in the settings of the different data exchange mechanisms offered by OP CUA, may heavily impact on the overall performance of the data exchanges between industrial applications (e.g. latency, round-trip time, delays and so on).

Current literature presents few papers dealing with the performance evaluation of OPC UA; most of them focus only on particular services and/or aspects of the OPC UA specification. For example in [2,3] performance evaluation is carried on considering only the security mechanisms and services provided by the OPC UA specifications. In [4,5] a lot of comparisons between OPC UA and previous specifications

* Corresponding author.
E-mail addresses: salvatore.cavalieri@dieei.unict.it (S. Cavalieri), chiacchio@dmi.unict.it (F. Chiacchio).

0920-5489/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.csi.2013.06.004

(COM- and XML-based) are available; although these comparisons are interesting, no useful information about the performances of the OPC UA alone can be achieved. In [6] some few results about the real measurements of delays introduced by OPC UA communication stack are shown; the results presented are very interesting but limited to few real scenarios and are not able to point out general considerations about OPC UA performances.

The aim of this paper is to deal with the performance evaluation of OPC UA, pointing out all its main data exchange mechanisms which could influence the client/server communication in industrial environments. Furthermore, their impact on the overall performances will be presented and discussed. Finally, useful considerations about OPC UA performances are derived from the results presented; these considerations are aimed to help the OPC UA final user to choose the right data exchange mechanisms and to set the relevant parameters, in order to improve the OPC UA performance.

Some of the analyses and results presented in this paper have been already published in proceedings of international conferences by one of the authors [7–9]; the content of this paper deeply deals with the performance evaluation of OPC UA, presenting other important results never published.

2. OPC UA overview

The OPC UA specifications are currently made up by 11 parts [1,4]. The OPC UA architecture models OPC UA Client and Server as interacting partners.

Client and Server applications use OPC UA Client and Server Application Programming Interface (API) to exchange data, respectively. OPC UA Client/Server API is an internal interface that isolates the Client/Server application code from an OPC UA Communication Stack. The OPC UA Communication Stack converts OPC UA Client/Server API calls into Messages and sends them through the underlying communication entity; on the other hand, each Message received from the underlying communication entity is delivered to the Client/Server application by the OPC UA Communication Stack.

Implementation of the OPC UA Communication Stack is not linked to any specific technology; this allows OPC UA to be mapped to future technologies as necessary, without negating the basic design. Two data encodings are currently defined: XML/text and UA Binary. In addition, two transport mappings are available: UA TCP and SOAP Web Services over HTTP. Clients and Servers that support multiple transports and encodings will allow the end users to make decisions about trade-offs between performance and XML Web Service compatibility at the time of deployment, rather than having these trade-offs determined by the OPC vendor at the time of product definition.

Fig. 1 shows the OPC UA Client architecture; a Client Application uses the OPC UA Client API to send OPC UA Service and Publishing Requests to OPC UA Server, and to receive OPC UA Service Response and Notification from the OPC UA Server. The OPC UA Communication Stack converts OPC UA Client API calls into Messages and sends them through the underlying communication entity to the Server; the OPC UA Communication Stack also receives Response and Notification Messages from the underlying communication entity and delivers them to the Client application through the OPC UA Client API.

Fig. 2 shows the OPC UA Server architecture. The Server Application is the code that implements the function of the Server. Real objects are physical or software objects that are accessible by the OPC UA Server or that it maintains internally; examples include physical devices and diagnostic counters. Particular objects, called Nodes, are used by the OPC UA Server to represent real objects, their definitions and their References; the set of Nodes is called AddressSpace. Nodes are accessible by Clients using OPC UA Services (interfaces and methods). Fig. 2 shows the Nodes in the AddressSpace, the references between them (drawn by arcs connecting the Nodes) and their relationships with the real objects.

Particular objects, called Monitored Items, can be created inside an OPC UA Server, as shown in Fig. 2; they are entities created by the OPC UA Client that monitor AddressSpace Nodes and their real-world counterparts, as described in Section 2.1. Monitor Items are created inside Subscriptions, shown in Fig. 2, which are the contexts of the data exchange between server and client, as described in Section 2.1.

To promote interoperability of Clients and Servers, the OPC UA AddressSpace is structured hierarchically with the top levels the same for all Servers. OPC UA Servers may subset the AddressSpace into Views to simplify Client access.

Like the OPC UA Client, the OPC UA Server uses the OPC UA Server API to send Response and Notification Messages to OPC UA Clients. The OPC UA Communication Stack receives Request Messages and Publish Requests from the OPC UA Client and delivers them to the OPC UA Server Application through the OPC UA Server API.

2.1. Client/Server data exchange

The simplest way for a Client and Server to exchange data is using the Read and Write OPC UA services, which allow an OPC UA Client to read and write one or more attributes of Nodes, maintained into the AddressSpace of the OPC UA Server; like most other services, the Read and Write services are optimised for bulk read/write operations and not for reading/writing single values.

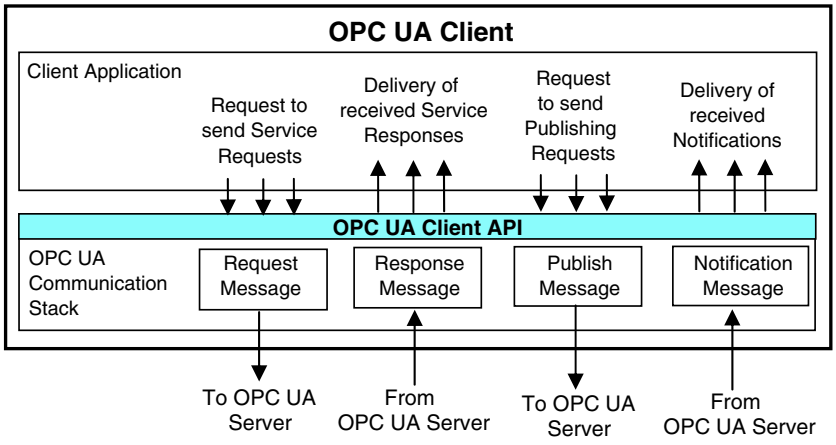

Fig. 1. OPC UA Client.

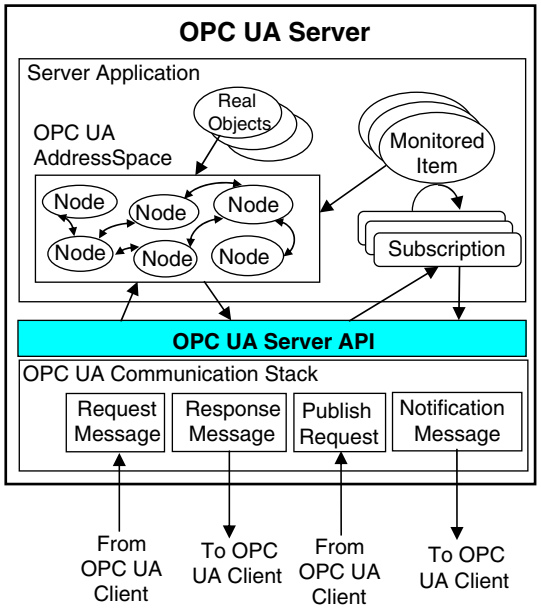

Fig. 2. OPC UA Server.

A different and more sophisticated way to access data is based on Subscriptions and Monitored Items; this is the preferred method for clients needing cyclic updates of variable values. Subscriptions and Monitored Items are put on the top of a Session level, which is a logical connection between an OPC UA Client and an OPC UA Server created in the context of a Secure Channel, described in the Section 2.2.

A Subscription is the context to exchange values on data changes, aggregates of data and events between the server and client; it requires a Session to transport the data to the client. Monitored Items can be created in a Subscription; they are entities in the OPC UA Server created by the OPC UA Client that monitor AddressSpace Nodes and their real-world counterparts. Three types of Monitored Items can be created. The first is used to subscribe for data changes of Variable Values; the second type of Monitored Item is used to subscribe for Events by defining an EventNotifier and a filter for the Event to be monitored. The third type of Monitored Item is used to subscribe for aggregated Values calculated based on current Variable Values in client-defined time intervals. Fig. 2 shows Monitored Items and Subscriptions; as can be seen each Monitored Item is related to Nodes into the AddressSpace and is bounded to a specific Subscription.

All Monitored Items have common settings, among which there are the sampling interval and the Monitored Item queue size. The sampling interval defines the rate at which the server checks Variable Values for changes or defines the time the aggregate get calculated. The Monitored Item queues are used to hold a certain pre-defined number of data produced by the Monitored Items, until the Subscription publishes them, as explained in the following. Fig. 3 shows a subscription containing the three kinds of Monitored Item described before; in the figure, the sampling intervals and the Monitored Item queues are shown.

In order to explain the data exchange based on Subscriptions and Monitored Items, please refer to Fig. 3 seen before. The figure points out the Publish Interval, which is one of the Subscription settings; the Publish Interval defines when the server clears the Monitored Item queues and conveys their contents into a Notification to be sent to the Client. Notifications will be sent to the Client by issuing the Publish service, as explained in the following.

Transmission of Notifications by OPC UA Server is triggered by Publish Requests sent by client. According to OPC UA specifications, a client must send a list of Publish Requests without expecting an immediate response; the server enqueues the Publish Requests until a Notification is ready for

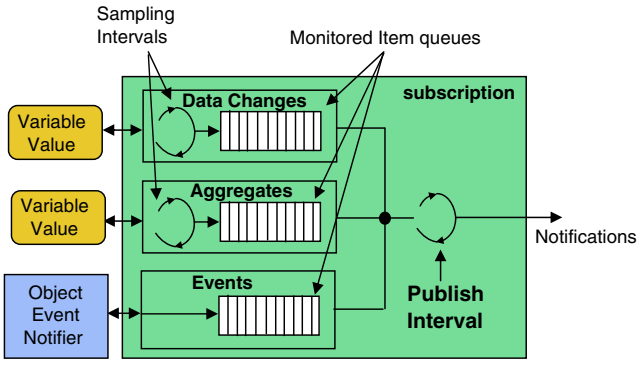

Fig. 3. Subscription.

sending to the client (according to the Publish Interval, as said before). When this occurs, the Notification is sent back to the client through a Publish Request response. The Publish Request is not bound to a specific Subscription and can be used by the server for all the Subscriptions running in the same Session context. To make sure that all Subscriptions can send a notification message at the same time, the client should make sure that there are more outstanding Publish Requests than active Subscriptions.

Fig. 4 depicts the exchange of Publish Request and Response between OPC UA Client and Server; as it can be seen, data exchange is realised within a Session, previously opened by client and server. The Session may contain several Subscriptions; for each of them, Notifications produced on the basis of the Publish Interval are waiting to be transmitted. For each Publish Request sent by the OPC UA Client, exactly one Notification is transmitted; it may belong to one of the current Subscriptions inside the Session.

2.2. Security model

OPC UA provides a security model, which includes security mechanisms allowing the authentication of Clients and Servers, the authentication of users, the integrity and confidentiality of their communications, and the verifiability of claims of functionality. Several parameters may be selected and may be set to personalise the security mechanisms to meet the security needs of a given installation. Furthermore, a minimum set of security Profiles that all OPC UA Servers support (even though they may not be used in all installations) has been defined.

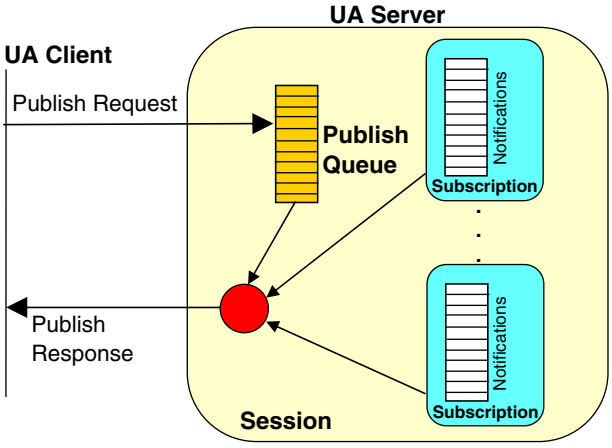

Fig. 4. Transmission of notifications by Publish Request.

Security relies on a Secure Communication channel that is active for the duration of the application Session and ensures the integrity of all Messages that are exchanged.

When a Session is established, the Client and Server applications negotiate a secure communications channel and exchange software Certificates that identify the Client and Server and the capabilities that they provide. Authority-generated software Certificates indicate the OPC UA Profiles that the applications implement and the OPC UA certification level reached for each Profile. Certificates issued by other organisations may also be exchanged during Session establishment.

The Server further authenticates the user and authorises subsequent requests to access Objects in the Server. Authorisation mechanisms, such as access control lists, are not specified by the OPC UA specification; they are application or system-specific.

OPC UA security allows to encrypt and sign Messages; encryption and signatures protect against disclosure of information and protect the integrity of Messages, respectively. OPC UA uses symmetric and asymmetric encryption to protect confidentiality as a security objective; asymmetric encryption is used for key agreement and symmetric encryption for securing all other messages sent between OPC UA applications. OPC UA uses symmetric and asymmetric signatures to address integrity as a security objective. The asymmetric signatures are used in the key agreement phase during the Secure Channel establishment; the symmetric signatures are applied to all other messages.

It's very important to point out that OPC UA specification doesn't make mandatory the use of certificates, digital signatures and data encryption. It's care of the final user of the OPC UA to evaluate when the choice of one or more of the previous security mechanisms is more appropriate; choice must be taken on the basis of the best trade-off between security requirements and the overall performance of the system, which may be influenced by certain security mechanisms as pointed out in this paper.

3. OPC UA performance

One of the main requirements for OPC UA is performance; OPC UA must scale from small embedded systems up to enterprise systems with different requirements regarding the speed and type of transferred data. In embedded systems, where smaller pieces of data must be transferred in short time intervals, the speed of the data transfer and minimal system load are the most important requirements. In enterprise systems, where structured data must be processed in a transaction- and event-based manner, the efficient handling of structured data is more important than the absolute speed of data transfer [4].

OPC UA features a very complex architecture made up by a very huge number of mechanisms, each of which can be enabled/disabled and can be personalised by setting particular parameters; for example, the previous section pointed out the complexity of the Subscription/Monitored Items mechanisms and the amount of parameters that can be set inside them (e.g. sampling interval, queue size and Publish Interval). Choice of the values of the parameters featuring each mechanism present in OPC UA, is under the responsibility of the final OPC UA user during the system configuration; each choice must be taken aiming to fulfil the constraints of the industrial application and the relevant requirements in terms of performance of the system.

For this reason, assessment of performance of OPC UA specifications seems very important in order to verify if and when the requirements of industrial applications are met by OPC UA architecture; furthermore, the impact of each mechanism of the OPC UA specifications (and the relevant values of parameters) on the overall performance of the system should be analysed very carefully. Results of this analysis may help the final user to evaluate if the choice of one of the foreseen mechanisms and the choice of the value for each of the relevant foreseen parameters are more appropriate to fulfil all the constraints and requirements of the industrial application.

Due to the huge number of mechanisms in OPC UA specifications, it's clear that performance evaluation should be preceded by an analysis aimed to point out the mechanisms of the OPC UA specifications which, more than others, could influence the behaviour of industrial applications using OPC UA to exchange information. The aim of the following subsections is to point out the main features of the OPC UA specification candidate to influence the relevant performance.

3.1. Security

The main question about performance evaluation of security aspects of OPC UA specifications is whether the OPC UA security model is efficient in data transfer.

OPC UA is used at different levels of the automation pyramid for different applications within the same environment. At the plant floor level, an OPC UA server may run in a controller providing data from field devices to OPC UA clients (e.g. HMIs, SCADA). On top of the plant floor at operation level, an OPC UA application may be a client collecting data from the server at the lower level, performing special calculations and generating alarms; an example is represented by an OPC UA client integrated in an ERP system, obtaining information about used devices in the plant floor (e.g. working hours) and creating a maintenance request.

For each application involving OPC UA, the trade-off between security and performance must be reached; at the very top level, security might be more important than performance since the corporate network is connected to the Internet. At the very bottom level, performance could be more important than security when data has to be acquired in a very fast and efficient way in order to control a production process.

Performance evaluation seems to play a very strategic role in order to reach the above-mentioned trade-off between performance and security. In particular, at least two different aspects of the security OPC UA model need to be investigated during performance evaluation.

The first is related to the use of the certificates and their verification operated by local or remote Certification Authorities (CAs) while opening a secure channel. In e-commerce environment a waiting time of 5–10 s until the Web server hosting a Web shop has validated the certificate of the customer, is very common and doesn't represent a very long time to purchase confirmation. However, 5–10 s can be a very long time interval for industrial applications, especially for devices located at the field level of the automation pyramid (e.g. applications in chemical or pharmaceutical industries, where very short waiting time could lead to serious problem). It's clear that many sessions in industrial applications may be characterised by very long duration, as they remain open for long period of time; for example, an operator workplace supervising a special area of a power plant can be connected to a server for 10 years without termination. Considering the data exchange inside sessions which remain open for very long periods, waiting times of tens of seconds to validate a certificate when the session is opened (at the start-up) are negligible. But, industrial applications are featured by a lot of applications which must be connected to a server for short period of time; furthermore these applications must access the server when needed without any delays. The most typical example of such applications are supervising and monitoring applications to manage faults or emergencies; these applications create a connection to the server to properly manage fault or emergency, only for the time period needed to resolve the problem. In those cases, any delay in the connection should be avoided.

The other aspect of OPC UA security which seems very important to investigate is relevant to the impact of data encryption/decryption and the digital signature of each message exchanged between OPC UA Client and Server. As known, encryption allows to achieve confidentiality in the data exchange, but in many applications at field device level, confidentiality isn't a strong requirement; for this reason, an analysis of the overhead introduced by encryption during data transfer should be performed in order to highlight if and when data encryption may not

be used (as it isn't mandatory according to the OPC UA specifications). The same considerations must be extended to the digital signatures foreseen in the OPC UA specifications (but not compulsory) and aimed to maintain integrity; also in this case a study of the overhead introduced by the signature of each message exchanged seems very important.

3.2. Transport protocols and encoding rules

As said, OPC UA may use the two different transport technologies: UA TCP and SOAP protocol; furthermore, both binary and XML encoding are currently available. Performance evaluation should take into account different transport protocols and encoding rules, comparing their impact on data exchange. In the paper, both UA TCP and SOAP transport protocols have been considered; due to the well-known performance of binary encoding, the XML encoding has not taken into account.

3.3. Subscription and Monitored Items

Subscription is the mechanism able to deliver information produced in a cyclic fashion. The previous section pointed out that the subscription is based on several parameters; among them, the Publish Interval seems to play an important role in the overall performance. As said, this parameter determines the time instants at which the Monitored Item queues (containing values coming from changes of variable values or from aggregates of variable values or from events), are emptied and a Notification is prepared to be sent to the Client; the Notifications produced are then pulled by the Client issuing Publish Requests. It's clear that a small value of the Publish Interval allows the client to receive fresh values, as the Monitored Item queues are emptied very soon, but requires a large amount of Publish Requests to be sent (reducing the available bandwidth in the underlying communication entity). On the other hand, greater values of Publish Interval lead to Notifications containing huge number of data (i.e. variable values, aggregates and events), some of which may be obsolete for the client; in this case Client is compelled to send low numbers of Publish Request.

On the basis of what said, the number of outstanding Publish Requests a client should maintain is another very critic parameter and could influence the overall performance of the system. Frequency of transmission of Publish Requests depends on the Publish Interval value, as said before, but it may be linked also to other events; for example, additional Publish Requests may be required if the latency of the network connection is very high. In any case, the number of outstanding Publish Requests maintained by each Client may strongly impact on the bandwidth utilisation, and could lead to bottlenecks in the client/server data exchange.

4. OPC UA model

OPC UA performance evaluation has been realised by the simulation of an ad-hoc model of the OPC UA client/server data exchange, instead of using a real implementation (based, for instance, on stacks and SDKs provided by the OPC Foundation). The main advantage offered by this choice is that the use of a model avoids the need to detail the internal mechanisms of the OPC UA specifications not directly involved in the aim of the performance evaluation, focusing only on certain mechanisms and their impact on performances. Furthermore, the use of a model allows achieving performance evaluation results not linked to a specific operating system, to a particular software development environment (i.e. to particular libraries) and to specific hardware architecture.

The model of the OPC UA client/server data exchange has been realised inside OMNeT++ framework [10]; other frameworks have been used to support the OPC UA model: the OpenSSL [11] and the INET framework [12]. OpenSSL libraries have been used to realise the main security mechanisms foreseen by OPC UA specifications; in particular, the encryption and decryption mechanisms have been realised using the Basic128RSA15. The INET framework has been used to realise the Point-to-Point Internet Protocol (PPP) at data link layer, through which the Client and Server data exchange was realised.

Fig. 5 shows the OPC UA model made up by a Client and Server Application exchanging data through several layers and using PPP for the communication. The first layer under the Client or Server Application is the Data Access which offers the set of OPC UA Services used in the model for the data exchange; then a Serialisation level realises the encoding of data. If secure mechanisms are enabled, the Security level performs the operations of security through signature (Sign) or through signature and encryption (Sign&Encrypt). Finally, Transport layer realises the transport services SOAP and UA TCP.

In more details, the main features implemented for each layer shown in Fig. 5, are:

- DataAccess. It implements all the main OPC UA services involved in the data exchange between client and server taken into account in the paper: data exchanges based on read/write services and those based on the use of Subscriptions/Monitored Items. In particular, the read() method has been implemented in order to realise the OPC UA Read service (see [1] Part 4-Services) to read the attributes of the variables exposed by an OPC UA Server. The CreateSubscription(), CreateMonitoredItem(), and Publish() methods have been implemented in order to realise the OPC UA services CreateSubscription, CreateMonitoredItems and Publish (see [1] Part 4-Services); these services allow to realise the data exchange based on the use of Subscriptions.
- Serialisation. It has been assumed to realise only the UA Binary encoding for the data exchange. For this reason, this layer implements all the methods needed to apply this encoding to each messages exchanged.
- Security. This layer ensures the transmission of the message in a secure mode through signature (Sign) or through signature and encryption (Sign&Encrypt). The operations performed are responsible for additional delays since any frame to transmit is manipulated in order to include the information needed by the destination node for its decoding. Moreover, at the beginning of the first transmission the secured communication context has to be open, as shown in Fig. 6. The main methods implemented are: GetEndPoints(), OpenSecureChannel(), CloseSecureChannel(), CreateSession(), ActivateSession(),

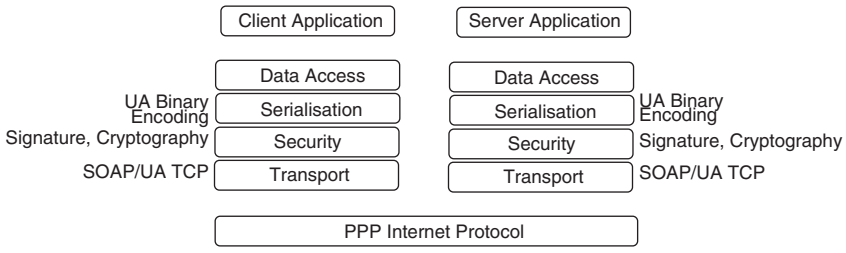

Fig. 5. OPC UA Client/Server model.

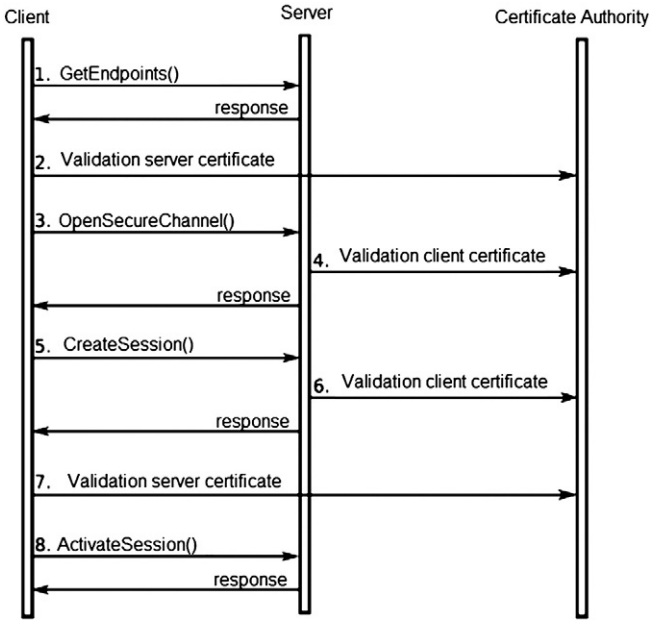

Fig. 6. Operations for the establishment of a communication context.

CloseSections(), and ValidateCertificate(). They realise the relevant OPC UA Services (see [1] Part 4-Services): GetEndpoints, OpenSecureChannel, CloseSecureChannel, CreateSession, ActivateSession and CloseSection; ValidateCertificate() realises all the procedures needed to validate the client and/or server security certificate through a local or remote certification authority. Verification of certificates by local and remote CAs has been modelled using real examples of CAs and deriving from them the average delays needed to perform verification; the values of these average delays have been used in the OPC UA model to represent the time spent for the verification of certificates by local or remote CAs, when requested during the simulation.

- Transport. This layer realises the transport services through UA TCP and SOAP/HTTP, used both in the OPC UA performance evaluation presented in the paper.

The implementation of the OPC UA model shown in Fig. 5 has been realised through the coding of a software library in C++ which has been integrated within OMNET++ and INET frameworks. For each layer of the stack of Fig. 5, three main type of files have been coded: (1) .ned, (2) .h and (3) .cc; .ned files are used by OMNET++ to instantiate the layers of the stack for the simulation, while .h and .cc are respectively the interface and the implementation of the C++ class inside in which the functionalities of the stack layers are coded. The Client and Server Applications have been realised by coding the type of files (1), (2) and (3).

The full model realised is available at [13], where detailed instructions are given in order to download, install and execute it.

4.1. Model validation

A great effort has been put to verify that the model behaved exactly as stated by the OPC UA specifications; this has been done executing all the methods implemented in the model realising OPC UA services, checking their behaviour inside the OMNeT++ framework (using the available tools), and verifying their full compliance with the OPC UA specifications.

Once each method implemented in the model has been validated as explained before, validation has been performed also in terms of performances. As said in the Introduction, literature presents few works about OPC UA performance evaluation. Among them, the Intel study presented in [6] presents some interesting real measurements of OPC UA performance in terms of round-trip delay; considering OPC UA Client/Server applications, round-trip time may be defined as the total response time between the instant at which a request to access (read/write) one or a set variables is issued by a client and the instant at which this access is realised by the server. The same scenarios used for the round-trip time measurements presented in [6], have been implemented in our model and the relevant performance evaluation has been achieved through the simulation of the model, as explained in the next section. Table 1 summarises the main results achieved, comparing the round-trip times when two different sizes of data blocks are transferred between client and server. As it can be seen, performances achieved by simulation of the OPC UA model are quite similar to that really measured inside the Intel study [6].

5. Performance evaluation

The main aim of the paper was to analyse the impact of the OPC UA mechanisms (e.g. security, transport, encryption subscriptions, monitored items) and the relevant parameters (e.g. signature, encryption, publish interval) onto the data exchange between OPC UA-based client and server applications. Both kinds of data exchange pointed out in Section 2.1 were considered, i.e. the one based on read/write services and the one based on Subscriptions/Monitored Items. Different evaluations have been carried on according to the kind of data exchange, as explained in the following.

As said in the previous sections, the most common data exchange occurs when a client application (e.g. SCADA) requires to a server application (e.g. an acquisition board) the (read and/or write) access to one or more variables; each access request may happen at unforeseeable time instants, generally driven by events linked to the specific industrial application. This data exchange may involve simple variables (e.g. an integer, a float) or complex data structures (e.g. made up by several bytes or a large set of variables). *Round-trip time* seems a suitable parameter to measure the performance of this kind of data exchange, because it points out the response time of the underlying communication system (including OPC UA stack) for each read/write request issued by a client. It's commonly used in literature, as done in [6], for instance. Definition of round-trip time has been given in Section 4.1.

The other kind of very common data exchange in industrial applications, happens when a client accesses to the values of variables updated in a periodic fashion; an example is a server which maintains values of a variable achieved by a sampling process (e.g. values of temperature obtained by sampling the original analogue signal) and a client which needs to read these values according to the same production (i.e. sampling) period. Fig. 7 shows a server producing values at a certain period T; a client application should receive each value produced within a deadline (set to the period T in the figure). In this case, the *delay* between the instant at which each value has been produced and the instant at which the client receives that value, seems to be of interest for performance evaluation. In fact, it's clear that the average delay for each variable gives a measurement of the efficiency of data exchange, pointing out the capability of the underlying communication system (including OPC UA stack) to deliver to the client each information periodically produced within the deadline. Again the kind of information involved in this scenario may be simple or complex data structures.

Table 1
Round-trip for data blocks at 1 Gbps.

Study	1 kbyte	2.5 Mbytes
Intel	1.2–1.4 ms	118–300 ms
OPC UA model	1.5–2.0 ms	225–420 ms

The next sections will point out the main results achieved in the performance evaluation carried on, mainly in terms of round-trip times for the read/write data exchange and delays when using subscriptions/monitored items.

6. Results

This section will present the main results of the performance evaluation of OPC UA, realised through the simulation of the model described in the Section 4, using OMNeT++ environment [10].

The performance evaluation focused on the three mechanisms which more than others seem to influence the OPC UA-based client/server data exchange; as shown in Section 3, they are the security, the transport and that based on Subscriptions/Monitored Items. Comparison of the effects produced on the OPC UA performance by the two different available encoding mechanisms (binary and XML) was avoided as it has been assumed to consider only the implementation of the binary encoding, due to its well-known best performance.

6.1. OPC UA security and transport mechanisms

The security aspects of OPC UA specifications which have been investigated, are the verification of certificates by a CA and the data encryption and signature of each message exchanged by OPC UA client and server applications. The following main scenarios have been considered:

(1) data exchange with no security mechanisms,
(2) use of the Secure Channel with no use of certificate (e.g. using passwords or other credentials),
(3) use of the Secure Channel with local verification of the certificates (i.e. operated by a local CA) and
(4) use of the Secure Channel with remote validation of the certificates (i.e. operated by a remote hierarchy of CAs).

For scenarios (2), (3) and (4), the following sub-cases have been considered: (a) no other security option used, (b) use of only digital signature for each message exchanged, and (c) use of both signature and data encryption for each message exchanged. About scenario (1), it's clear that no sub-cases must be defined.

Combining scenarios (2), (3) and (4) with the previous three sub-cases (a), (b) and (c), and including scenario (1) alone, 10 different scenarios have been achieved. In the following, scenario (1) (data exchange with no security mechanisms) will be simply named scenario number 1, as it is not featured by any sub-cases; the other scenarios will be indicated using a number (ranging from 2 to 4) a dot and a letter (a, b and c). The number refers to one of the scenarios (2), (3) and (4) and the letter refers to one of the 3 sub-cases; for example scenario 4.a means use of the Secure Channel with remote validation of the certificates, and no other security option used.

For each of the 10 scenarios, it has been assumed to realise the data exchange between a couple of OPC UA client and server at different available bandwidths: 2, 5 and 10 Mbps; only results related to a 2 Mbps bandwidth will be shown in the following, as they were quite similar. Data exchange between client and server applications has been assumed to be realised through Read service (see [1] Part 4-Services).

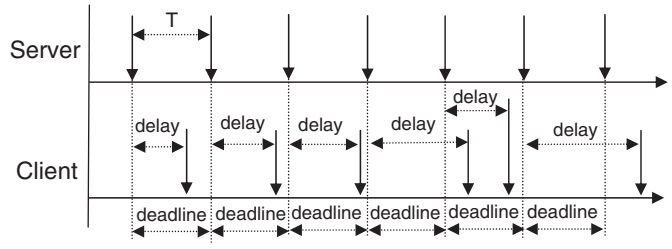

Fig. 7. Delay between each information produced by a server and delivered to a client.

A first set of simulations has been carried on in order to highlight the influence of the security mechanisms on the times needed to open and activate a secure session. The four main scenarios (1), (2), (3) and (4), without the sub-cases (as they have no influence on the activation of a session), have been considered; both the two transport protocols (UA TCP and SOAP) have been considered. Table 2 shows the times needed to activate a session considering UA TCP and SOAP. In scenario (4) the times needed to activate a session are very huge (14.0 s for UA TCP and 15.9 s for the SOAP) only due to the need to use of remote certification authorities. In every scenario, use of UA TCP instead of SOAP leads to a little save in time.

Performance evaluation has been carried on also to investigate the influence of digital signature, data encryption and transport protocol on the round-trip times; evaluation of the round-trip times has been achieved considering different sizes (in bytes) of the set of variables read from the server. The first main result achieved was that the choice of scenario 1 (no security) and UA TCP protocol options allows the best performances in terms of round-trip times (i.e. the lowest round-trip time values have been achieved). Table 3 gives the values of the Round-trip time related to the scenario 1 and UA TCP, for the different sizes of variables taken into account.

In the following, the round-trip time evaluations achieved for the other scenarios will be presented. It has been assumed to represent these values normalised to those shown in Table 3; in this way the reader can easily compare each of the scenario considered with that featuring the best performance (i.e. scenario 1 using UA TCP).

Fig. 8 shows the round-trip times relevant to scenario 1 with SOAP protocol; abscissa of the figure is the size (in bytes) of the set of variables read from the server. The curve in the figure is obtained normalising the values of the round-trip time to those achieved considering scenario 1 (no security) and UA TCP, as said before. Fig. 8 points out that when no security mechanisms are considered, use of SOAP instead of UA TCP produces higher round-trip times. This happens mainly for small sizes of variables; the behaviour of UA TCP and SOAP tends to converge for huge sizes of data read.

Fig. 9 compares the round-trip times achieved considering scenario 2.c with UA TCP and SOAP protocols; again, each curve of the figure is obtained normalising the values of the round-trip time to those achieved considering scenario 1 (no security) and UA TCP. So curve labelled with "2.c UA TCP" refers to the round-trip times concerning scenario 2.c and UA TCP, normalised to the values achieved considering scenario 1 and UA TCP; curve "2.c SOAP" refers to the round-trip times considering scenario 2.c and SOAP normalised to the values achieved considering scenario 1 and UATCP. Figs. 10 and 11 compare the round-trip times considering scenarios 3.c and 4.c and both UA TCP and SOAP protocols. Again the round-trip values are normalised as explained before and the abscissa refers to the size of data read from the server. Figs. 9–11 point out that the use of UA TCP also in presence of the security mechanisms leads to lower round-trip times, compared with SOAP-based data exchange. Again, this mainly occurs with small sizes of variables exchanged; for higher sizes, the performance of the UA TCP and SOAP tends to converge, also in the presence of security mechanisms.

The last set of measures about security here shown, are relevant to the comparison of the influence of the different configurations which may be chosen inside a secure channel; these configurations are relevant

Table 2
Times needed to activate a Session.

Scenario	UA TCP	SOAP
(1)-No security mechanisms	0.054 s	0.1 s
(2)-Secure Channel with no use of certificate	0.16 s	0.28 s
(3)-Secure Channel with local CA verification	0.31 s	0.43 s
(4)-Secure Channel with remote CA verification	14.0 s	15.9 s

Table 3
Round-trip time related to scenario 1 and UA TCP.

Bytes	4	1024	2048	4096	8192	16384	32768
Round-trip time (s)	0.003	0.019	0.036	0.068	0.1	0.19	0.39

to the sub-cases (a), (b) and (c). The round-trip times have been measured again, considering only the UA TCP transport mechanism and the three scenarios (2), (3) and (4) which feature the use of secure channel. The results achieved for the three scenarios are quite similar, so only those relevant to one of them (the fourth) will be presented in the following. Fig. 12 points out the round-trip times considering only the UA TCP mechanism and scenarios 4.a, 4.b and 4.c; as done before, these values have been normalised to those achieved considering scenario 1 (no security) and UA TCP (shown in Table 3). As can be seen from the figure, the influence of the digital signature and data encryption mechanisms is very huge for small size of variables exchanged; performances of the different scenarios converge when variables increase in size.

6.2. OPC UA subscription mechanisms

Influence of settings related to subscription/monitored items mechanisms on the overall OPC UA performance has been also investigated; this subsection will present the main results achieved.

It has been assumed that the server has to publish 3 sets of variables; 5 variables for each set have been considered. Each variable is periodically updated, i.e. a new value is produced at a certain period, as shown in Fig. 7. All the variables belonging to a set has a common production period; the following periods have been considered for the three sets: 5, 10 and 15 ms. The size of each variable was fixed to 4 bytes.

A Session on the top of a secure channel has been considered for the server. It has been used to instantiate only one Subscription reserved for the transmission of the values of the 3 sets of variables described above. A Monitored Item has been associated to each variable and configured to subscribe for data changes; the sampling interval (see Fig. 3) has been set equal to the relevant production period (i.e. 5, 10 or 15 ms). For all the Monitored Items, the size of the Monitored Item queue (see Fig. 3) has been fixed in order to avoid overflow, i.e. loss of values; this size has been determined running several simulations, analysing the queue occupation and choosing the suitable value able to avoid overflow.

Different values of Publish Interval (see again Fig. 3) have been considered during the performance evaluation, ranging from 5 ms to 250 ms.

All the values produced by the Monitored Items are transmitted to the client inside Notifications, on the basis of explicit requests issued by the client through the Publish Request service (see [1], part 4, and Fig. 4). The time interval according to which client issues a Publish Request has been fixed to the 80% of the Publish Interval.

Only UA TCP with binary encoding has been considered and the scenario 4.c seen before has been assumed for the secure channel.

Different bit rates have been considered for the client/server data exchange based on Publish Request; in particular the following values have been taken into account: 2, 5 and 10 Mbps. In order to take into account the realistic presence of other data exchanges different from that previously described, for each available bandwidth it has been assumed that the 30% was reserved to other kinds of information flow.

For each value produced by a Monitored Item, the delay has been measured; according to what has been said in Section 5, delay has been defined as the time interval between the instant at which each value is enqueued in the Monitored Item queue (due to a data change) and the instant at which the client receives the Notification containing this value. As said in Section 2.1, a Notification delivered to a client is made up by all the values contained in a same Monitored Item queue; these values will be featured by different delays as their arrival times in the queue are clearly different. For this reason the lowest delay (relevant to the last value enqueued), the highest delay (relevant to the first value enqueued) and the average delay have been evaluated and updated for each Notification delivered to the client. Results presented in this section will point out these three values.

Fig. 13 shows the delay versus the Publish Interval, considering the Monitored Item featuring a sampling interval of 5 ms and an available bandwidth of 2 Mbps. The delay values have been normalised to the sampling interval; this means that a value of 1 corresponds to a delay equal to the sampling interval. Figs. 14 and 15 have a similar content, but they refer to Monitored Item with sampling interval of 10 and 15 ms, respectively; available bandwidth is always 2 Mbps. As can be seen the trends of the delay are quite similar for the three scenarios. Limited values of delay are guaranteed by low values of Publish Interval; when Publish Interval increases, delay may assume huge values.

Figs. 16, 17 and 18 show the delay versus the Publish Interval, considering the Monitored Item featuring a sampling interval of 5, 10 and 15 ms, respectively; in this case the available bandwidth is 10 Mbps. Again, the delay values have been normalised to the sampling interval. Also in this case, the trends of the delay are quite similar for the three scenarios; limited values of delay are guaranteed only by low values of Publish Interval.

Analysis of these curves could lead to the suggestions that use of low values of Publish Interval should be always encouraged; but this could not be true. In fact, low values of Publish Interval means a high frequency transmission of Publish Requests by client, i.e. overload on the transmission medium (as pointed out in Section 3.3). In order to achieve a complete view, the impact of low values of Publish Interval on the bandwidth utilisation must be evaluated. Figs. 19 and 20 show bandwidth utilisation versus the Publish Interval, considering a total bandwidth of 2 and 10 Mbps, respectively. All the values of bandwidth utilisation are above the 30% value, as it has been assumed that the data exchange based on Monitored Items is allowed to use only the 70% of the available

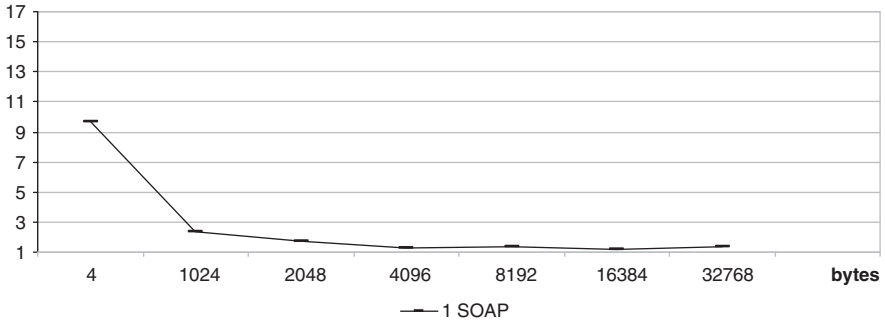

Fig. 8. Round-trip time, scenario 1 with SOAP.

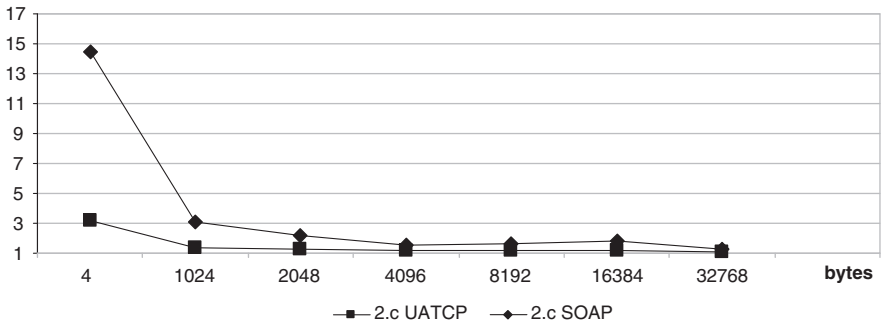

Fig. 9. Round-trip time, scenario 2.c with UA TCP and SOAP.

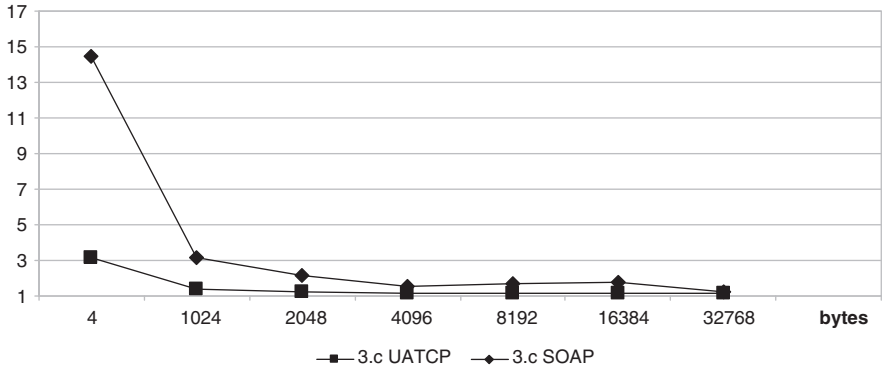

Fig. 10. Round-trip time, scenario 3.c with UA TCP and SOAP.

bandwidth; this means that the 30% is used to realise other type of information flow. As it can be seen from the figures, low values of Publish Interval lead to a very high bandwidth occupation; this is more evident considering Fig. 19, due to the low value of total available bandwidth.

7. Tuning OPC UA performance

The results presented in the previous section are enough to point out some considerations about the tuning of OPC UA settings in order to speed-up the relevant performances. As seen, the main results of the paper focused on the impact of OPC UA Security and Transport Protocol on the Client/Server Data Exchange based on Read/Write Services; the paper presented evaluation of round-trip times for the Read Service, but similar results may be expected for the Write Service, as their behaviour is quite similar. Furthermore, the paper presented evaluation of the effect of Subscriptions/Monitored Items mechanisms on the Client/Server data exchange.

The massive use of all the mechanisms taken into account in the paper in real OPC UA-based applications, gives an extreme importance to the analyses presented in the previous section and to the relevant considerations pointed out in the following.

7.1. OPC UA security mechanisms

OPC UA specification doesn't make mandatory the use of certificates, digital signatures and data encryption; this improves the strategic role of the performance evaluation here presented aimed to highlight the relevant overload introduced. Results achieved may help the final user of the OPC UA to evaluate when the choice of one or more of the previous security items is more appropriate.

In particular, the main considerations about the impact of security on OPC UA performances may be summarised as follows:

- Impact on the times needed to activate a section. Use of remote certification authorities to verify client and server certificates introduces huge delays and should be avoided.

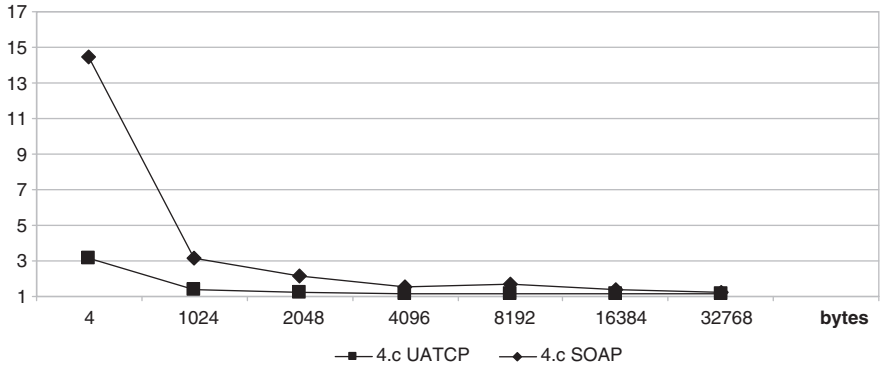

Fig. 11. Round-trip time, scenario 4.c with UA TCP and SOAP.

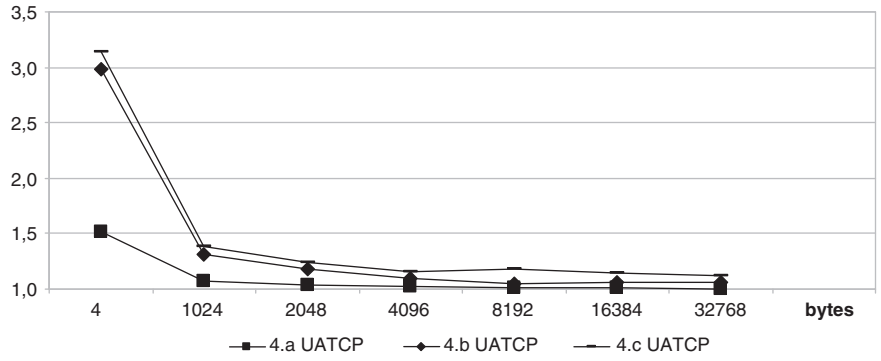

Fig. 12. Round-trip time, scenarios 4.a, b, and c with UA TCP.

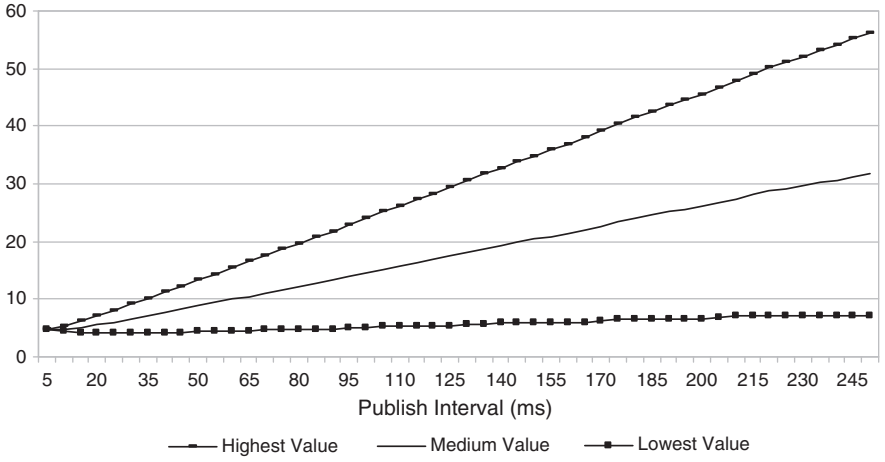

Fig. 13. Delay/sampling interval (5 ms) at 2 Mpbs.

- Impact on round-trip times. Security plays a strong influence on the round-trip times; the lowest values of round-trip times have been achieved in a scenario featuring the lack of security mechanisms, as shown in Table 3. The influence of the digital signature and data encryption mechanisms is very huge for small sizes of variables exchanged; exchange of bulk of data using read/write services is strongly encouraged when security options have been chosen.

7.2. OPC UA transport protocol

Choice of UA TCP or SOAP transport protocols should be subjected to the following considerations derived from the results presented in the previous section:

- Impact on the times needed to activate a section. The times needed to activate a session may be reduced if UA TCP is adopted instead of SOAP.

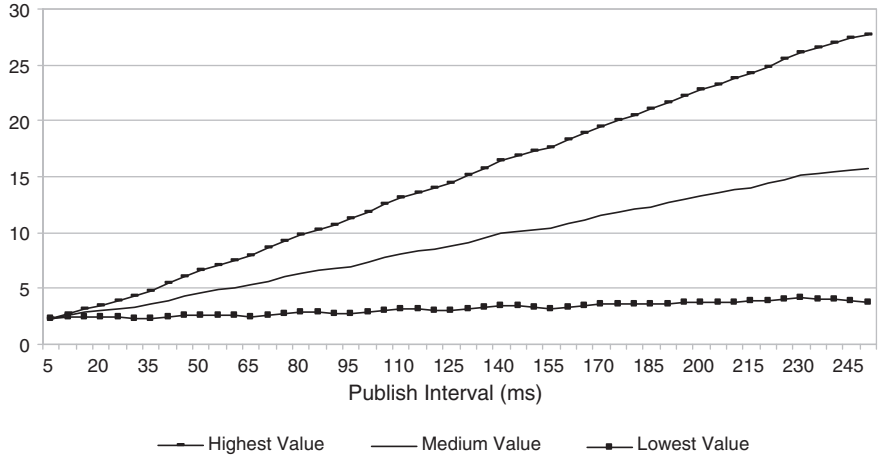

Fig. 14. Delay/sampling interval (10 ms) at 2 Mpbs.

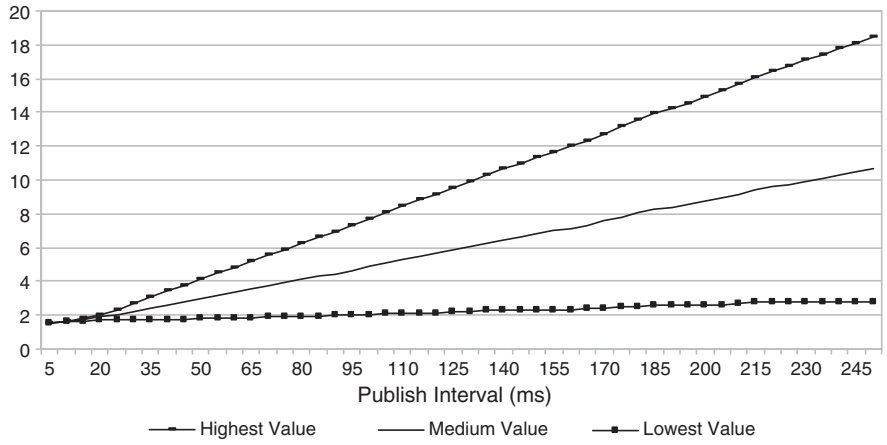

Fig. 15. Delay/sampling interval (15 ms) at 2 Mpbs.

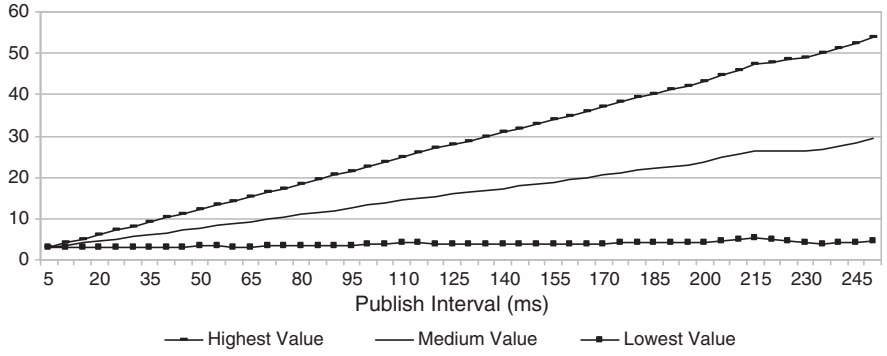

Fig. 16. Delay/sampling interval (5 ms) at 10 Mpbs.

- Impact on round-trip times. UA TCP protocol options allowed the best performances in terms of round-trip times, mainly with small size of data exchanged. When the size of variables exchanged increases, the performance of the UA TCP and SOAP tends to converge.

7.3. OPC UA subscription mechanisms

Data Exchange in OPC UA based on Subscriptions/Monitored Items features the need to set some parameters; among them the Publish Interval seem responsible to heavily influence the Client/Server data exchange, therefore a good tuning could really improve the relevant performance. On the basis of the results presented in the previous section, the following considerations could be pointed out about the settings inside a subscription:

- Impact on delay. For each message exchanged, limited values of delay (see Fig. 7 for the definition of delay) are guaranteed by low values of Publish Interval; when Publish Interval increases, delay may assume huge values not compatible with the automation applications on the top of OPC UA client/server.
- Impact on bandwidth. The Publish Interval has also an impact on the bandwidth utilisation of the communication network connecting client and server. This is mainly due to the fact that the frequency at which the OPC UA client has to produce and transmit Publish Requests must be strictly linked to the Publish Interval values; lower values of

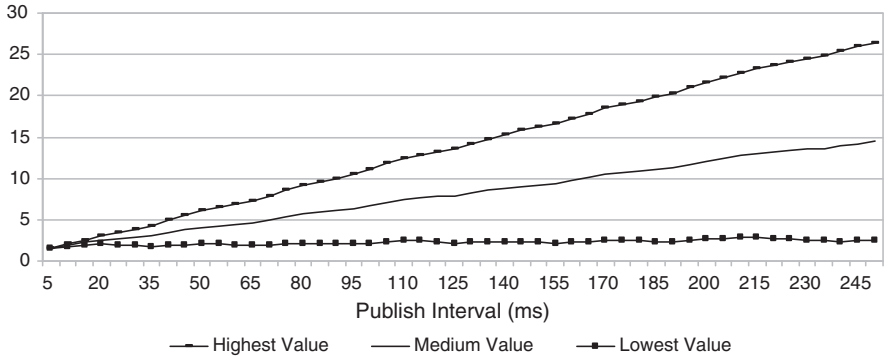

Fig. 17. Delay/sampling interval (10 ms) at 10 Mpbs.

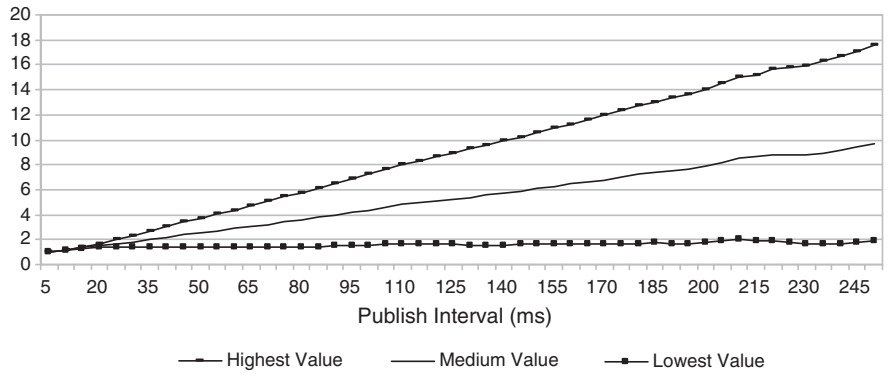

Fig. 18. Delay/sampling interval (15 ms) at 10 Mpbs.

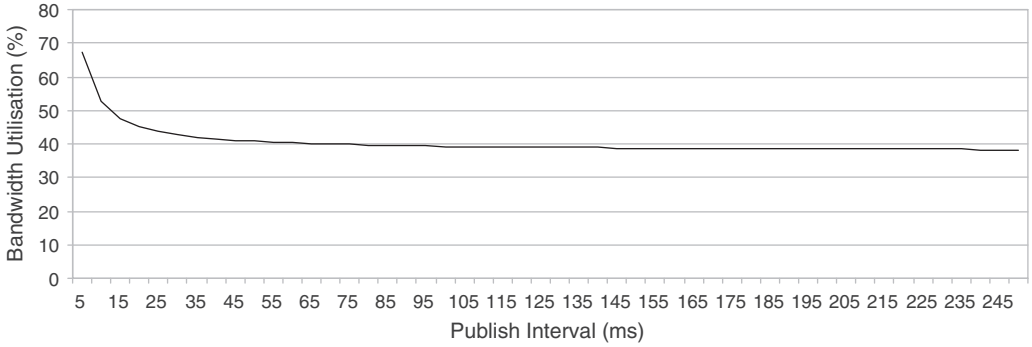

Fig. 19. Bandwidth utilisation at 2 Mpbs.

Publish Interval must correspond to higher values of frequency of transmission of Publish Requests. If this didn't happen, Notification messages produced at each Production Interval may be lost. On the basis of the results achieved, it's clear that low values of Publish Interval lead to a very high bandwidth occupation.

The two opposite effects of the Publish Interval on the delay of information sent to the client and on the bandwidth utilisation, make very critical the choice of Publish Interval. This choice must be subjected to a very difficult trade-off between the need to increase as much as possible the Publish Interval (in order to limit the bandwidth utilisation) on one hand and the need to reduce the Publish Interval on the other (in order to limit the delay of each information transmitted).

Detection of a suitable value of Publish Interval able to reach this trade-off can be realised in the case performance evaluations (like those depicted by Figs. 13–20) were available. For example, comparing Fig. 19 with Figs. 13–15, it's clear that values of Publish Interval close to 35 ms, represent a good trade-off between the need to receive (on the client side) fresh values of variables and the need to maintain suitable percentage of available bandwidth.

In practice, when configuring a real OPC UA Server for a particular control application scenario, a performance evaluation like the one here presented could be not available. In this case, choice of Publish Interval should be based on the considerations achieved from the main results presented in this paper. In particular, choice of low or high values of Publish Interval (where the meaning of low and high must be derived from its comparison with the sampling intervals of the variables to be read), must take into account the knowledge that use of high values of Publish Interval leads to high delays; on the other hand, low values of Publish Interval reduce the average delay but may increase the bandwidth utilisation.

8. Final remarks

The paper has presented the OPC UA specifications, pointing out some features which may have an impact on the overall performance

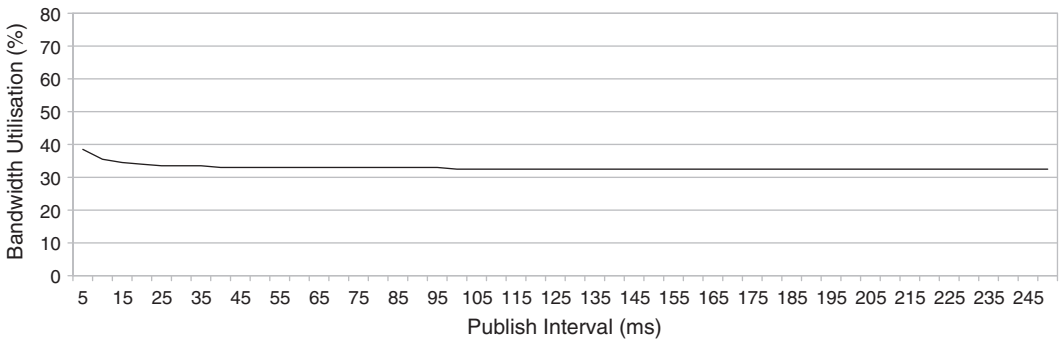

Fig. 20. Bandwidth utilisation at 10 Mpbs.

of the client/server data exchange. The paper has then proposed some measurement parameters to be considered in the performance evaluation. Finally, the main results achieved during performance evaluation have been presented and discussed. Results pointed out that some OPC UA parameters (e.g. Publish Interval) are very critical and their setting is very difficult because it's based on deep and technical considerations which require high level of expertise.

References

[1] OPC Foundation, "OPC Unified Architecture Specification", Parts 1–11. , available at www.opcfoundation.org2009.
[2] A. Braune, S. Henning, S. Hegler, Evaluation of OPC UA secure communication in web browser applications, Proceedings IEEE International Conference on Industrial Informatics (INDIN 2008), Daejeon, Korea, 2008, pp. 1660–1665.
[3] O. Post, J. Deppala, H. Koivisto, The performance of OPC UA Security Model at field device level, Proceedings ICINCO, 2009, pp. 337–341.
[4] W. Mahnke, S. Leitner, M. Damm, OPC Unified Architecture, Springer, 2009. (ISBN 978-3-540-68898-3).
[5] J. Lange, F. Iwanitz, T.J. Burke, OPC from Data Access to Unified Architecture, OPC Foundation, Softing, 2010.
[6] Intel, Reducing product development effort for OPC unified architecture. , available at http://download.intel.com/platforms/applied/indpc/OPC_Unified_Arch_CS.pdf.
[7] S. Cavalieri, Impact of OPC UA specification on industrial applications, Systemics and Informatics World Network 10 (2010) 179–187, (ISSN 2044–7272).
[8] S. Cavalieri, G. Cutuli, Performance evaluation of OPC UA, Proceedings 15th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2010), Bilbao, Spain, September 13–16 2010, pp. 1–8.
[9] S. Cavalieri, G. Cutuli, S. Monteleone, Evaluating impact of security on OPC UA performance, Proceedings 3rd IEEE International Conference on Human System Interaction (HSI 2010), Rzeszów (Poland), May 13–15 2010, pp. 689–694.
[10] www.omnetpp.org.
[11] www.openssl.org.
[12] http://inet.omnetpp.org/.
[13] http://www.dmi.unict.it/~chiacchio/projects/opc_ua_sim/.

Salvatore Cavalieri was born in Catania (Italy) in 1965. He received his "laurea" degree in Electronic Engineering from the University of Catania in 1989. In 1993 and 1995, he received a PhD in Electronic and Computer Science Engineering, and a post-PhD in Electric Engineering from the same University. Currently he is Full Professor of Computer Science at the University of Catania, Department of Electrical Electronic and Computer Engineering. His research areas are in neural networks, distributed systems, Grid/Cloud computing, real-time scheduling, process control oriented, industrial informatic and wireless communication protocols. In the area of process control and communication protocols, he served as member of the IEC SC65C WG6 Fieldbus standardisation committee; he is honorary member of Fieldbus Foundation Italy and Profibus Network Italy Consortium (a regional association of Profibus International, www.profibus.com). Since 2007 he is a Scientific Partner of KNX organisation (www.knx.org); since 2009 he is member of OPC foundation (www.opcfoundation.org).

Ferdinando Chiacchio was born in Acireale (Italy) in 1981. He received his "laurea" degree in Computer Engineering from the University of Catania in 2005. In 2010, he received a PhD in Mathematics Applied to the Engineering from the same University. Currently he is a Researcher of Computer Science at the University of Catania, Department of Electrical Electronic and Computer Engineering. His research areas concern reliability, performability, communication protocols for home and industrial control and automation, HPC computing and immunomics.