Evaluating Impact of Security on OPC UA Performance

Salvatore Cavalieri, Giovanni Cutuli and Salvatore Monteleone
University of Catania, Faculty of Engineering,
Department of Computer Sciences and Telecommunications Engineering (DIIT),
Viale A.Doria, 6 - 95125, Catania, Italy
Salvatore.Cavalieri@diit.unict.it

Abstract. **The widespread use of OPC UA specification in client/server data exchange in industrial environment introduces many benefits as it allows to keep open the market of the industrial applications, due to the presence of a standard, worldwide and vendor-independent specification. On the other hand, the use of a software infrastructures (complex in the case of OPC UA) between industrial applications and devices (e.g. mainframes, PLC, microcontroller), may impact on the overall performance of the communication between industrial applications. The aim of this paper is to deal with the performance evaluation of OPC UA, pointing out all the main features which could influence performance in the client/server exchange of information. On the basis of the considerations about performance of OPC UA specification, outlined in the paper as said before, some results of a performance evaluation carried on by the authors will be presented; it will focus on the impact of OPC UA security mechanisms on the relevant performance.**

***Keywords*: OPC UA, Security, Performance Evaluation, SCADA.**

I. Introduction

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

Definition of OPC specifications started ten years ago as an initiative of a handful of companies to simplify and to standardise data exchange between software applications; OPC specifications have been initially conceived merely to replace proprietary communication drivers for interfacing Supervisory Control and Data Acquisition (SCADA) systems and visualization programs to the process peripherals. The rapid success of OPC specifications was due to the rapid increase in the use of Windows PCs and the choice of Microsoft's DCOM as the technological basis for the OPC specifications, as DCOM was found on every Windows PC. Exactly this point, however, at the same time 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. Besides DCOM's long response times and inaccurate messages for connection interruptions, the need to disable security measures when configuring the DCOM settings was also criticised.

When XML and Web Services technologies have been 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 enables applications to communicate independently of the manufacturer and platform.

Today, the OPC Foundation has introduced the OPC UA standard which is based on a service-oriented approach. By defining service-oriented and platform-independent OPC specifications, the OPC Foundation has created new, 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 will not take the place of the OPC specifications that exist today, but supplement them; OPC UA expands the current OPC technology by significant functional aspects.

Based on XML and Web services, OPC UA allows the secure and reliable transport of raw data and pre-processed information though an interoperability platform and unifies the use of different Data Access (DA), Alarms&Events (AE) and Historical Data Access (HDA) servers and clients for vertical and horizontal data exchange. OPC UA servers provide access to current and historical data as well as events, such as alarms, value changes or the result of a program call. Where previously up to three different OPC servers – DA, AE and HDA – with different semantics used to be necessary to acquire the current value of a temperature sensor, the event of excess temperature and the historical average temperature, only one OPC server, the UA server, is needed with OPC UA specification.

OPC UA servers can be varied and scaled in their scope of functions, size, performance and the platforms they support. The server properties are specified in a profile and can be queried by the client. 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.

A key feature of the OPC UA specification is the definition of the UA 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 authorization within OPC communication relationships. For this purpose, the OPC UA specification uses various Web service standards such as WS-Security, WS-Trust or WS-SecureConversation. The scope of security measures that is to be supported by an OPC UA server is scalable and can be queried by the client. Security rules range from password authentication and the exchange of digital signatures through to full encryption of the OPC messages exchanged.

OPC messages are by default transmitted as SOAP messages, i.e. as XML text, between the UA services. As coding and decoding into the text format takes time and the overhead of textual representation is considerable, the OPC UA specification alternatively defines a UA binary format for data transfer via TCP/IP.

Due to the current features of the OPC UA specification, SCADA, PLC/PC-based controls and MES systems are unthinkable today without an OPC UA interface. Nowadays OPC UA plays a very dominant role in industrial applications as the most part of the data exchanges between client/server industrial applications are currently based on this specification. This introduces benefits as allows to keep open the market of the industrial applications, due to the presence of a standard, worldwide and vendor-independent specification. On the other hand the use of software infrastructures (complex in the case of OPC UA) between industrial applications and devices (e.g. mainframes, PLC, microcontroller), may impact on the overall performance of the communication between industrial applications.

Current literature presents few papers dealing with performance evaluation of OPC UA; most of them focus only on particular services and/or aspects of the OPC UA specification. For example in [1][2] performance evaluation is carried on considering only the security mechanisms and services provided by the OPC UA specification.

The aim of this paper is to deal with the performance evaluation of OPC UA, pointing out all the main features which could influence performance in the client/server exchange of information.

On the basis of the considerations about performance of OPC UA specification, outlined in the paper as said before, some results of a performance evaluation carried on by the authors will be presented; it will focus on the impact of OPC UA security mechanisms on the relevant performance.

II. OPC UA OVERVIEW

The OPC UA Specification comprises 13 parts [3][4]. Parts 1 to 7 constitute the core specification which defines the address space and the UA services. Parts 8 through 11 apply these core capabilities to specific types of access previously addressed by separate OPC COM specifications, such as Data Access (DA), Alarms and Events (A&E) and Historical Data Access (HDA). Part 12 describes Discovery mechanisms for OPC UA and Part 13 describes ways of aggregating data.

OPC UA defines a common infrastructure model to facilitate the information exchange between industrial processes, including Field Devices, Control Systems, SCADA Systems, Manufacturing Execution Systems (MES) and Enterprise Resource Planning Systems (ERP). This infrastructure is mainly made up by: an information model (to represent structure, behaviour and semantics), the message model (to interact between applications), the communication model (to transfer data), and the conformance model (to guarantee interoperability between systems).

OPC UA is a platform-independent standard through which various kinds of systems and devices can communicate by sending Messages between Clients and Servers over various types of networks. It supports robust, secure communication that assures the identity of Clients and Servers and resists attacks.

OPC UA defines sets of Services that Servers may provide, and individual Servers specify to Clients what Service sets they support. Information is conveyed using OPC UA-defined and vendor-defined data types, and Servers define object models that Clients can dynamically discover. Servers can provide access to both current and historical data, as well as Alarms and Events to notify Clients of important changes.

OPC UA is designed to provide robustness of published data; for example, OPC UA provides mechanisms for Clients to quickly detect and recover from communication failures associated with these transfers without having to wait for long timeouts provided by the underlying protocols.

To promote interoperability, OPC UA defines subsets, called Profiles, to which Servers may claim conformance; Clients can discover the Profiles of a Server, and tailor their interactions with that Server based on the Profiles.

The OPC UA specifications are layered to isolate the core design from the underlying computing technology and network transport. 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 defined: 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 tradeoffs between performance and XML Web service compatibility at the time of deployment, rather than having these tradeoffs determined by the OPC vendor at the time of product definition.

In the following subsections the main features of the OPC UA specification will be given; particular emphasis will be given to the OPC UA aspects influencing its performance, in order to help the reader to better understand the other sections of the paper, dealing with performance evaluation.

A. *Information Model*

The OPC UA architecture models OPC UA Clients and Servers as interacting partners; each Client may interact concurrently with one or more Servers, and each Server may interact concurrently with one or more Clients. An application may combine Server and Client components to allow interaction with other Servers and Clients.

A Client Application uses the OPC UA Client API to send and receive OPC UA Service requests and responses to/from the OPC UA Server. The "OPC UA Client API" is an internal interface that isolates the Client application code from an OPC UA Communication Stack. The OPC UA Communication Stack converts OPC UA Client API calls into Messages and sends them through the underlying communications entity to the Server at the request of the Client application. The OPC UA Communication Stack also receives response and Notification Messages (see subsections B and C for a definition of Notifications) from the underlying communications entity and delivers them to the Client application through the OPC UA Client API. Figure 1 shows the OPC UA Client architecture.

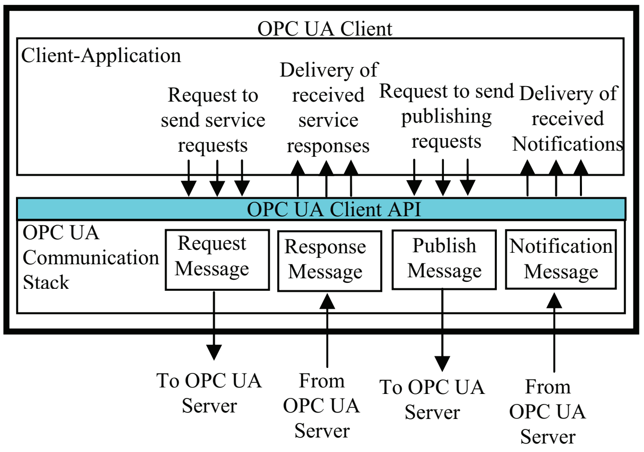

Fig. 1. OPC UA Client

Figure 2 shows the OPC UA Server architecture. The OPC UA 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 diagnostics counters. OPC UA Server uses the "OPC UA Server API" to send and receive OPC UA Messages from OPC UA Clients. As said for the client, the "OPC UA Server API" is an internal interface that isolates the Server application code from an OPC UA Communication Stack.

The set of Objects and related information that the OPC UA Server makes available to Clients is referred to as its AddressSpace, which is a set of Nodes accessible by Clients using OPC UA Services (interfaces and methods). Nodes in the AddressSpace are used to represent real objects, their definitions and their References to each other. OPC UA Servers may subset the AddressSpace into Views to simplify Client access. To promote interoperability of Clients and Servers, the OPC UA AddressSpace is structured hierarchically with the top levels the same for all Servers.

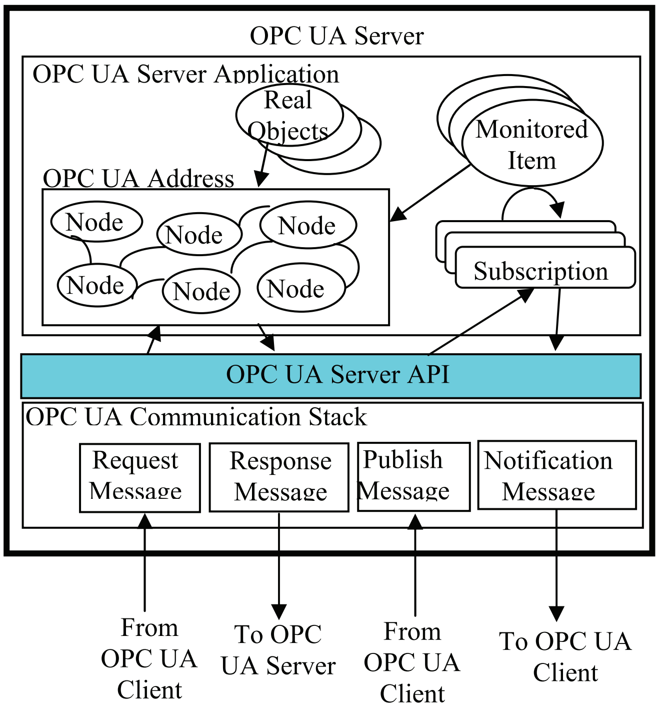

Fig. 2. OPC UA Server

B. *Client/Server Communication*

Client/Server communication is realised by OPC UA Services, which are methods used by an OPC UA Client to access the data of the Information Model provided by an OPC UA Server.

Definition of OPC UA Services is independent from the transport protocol and the programming environment that is used to develop an OPC UA application. This implies an abstract definition of the Services; this abstract definition can be applied to different transport mechanisms defining the representation of the Services on the wire, and to different implementations of the transport mechanisms in OPC UA Stacks based on the abstract UA Service definition.

OPC UA Services are not stateless and cannot be called without establishing a communication context on different levels. For this reason a lot of Services are used only to create, maintain, and modify these different levels of communication context. A communication context is made up by a stack of three levels: Monitored Item, Subscription and Session. All these three levels are put on the top of a Secure Channel level, described in the subsection D.

The Session is a logical connection between Clients and Servers created on the top of and in the context of a Secure Channel. The lifetime of a Session is independent of the Secure Channel and another Secure Channel can be assigned to the Session. Servers may limit the number of concurrent Sessions based on resource availability, licensing restrictions or other constraints. Each Session is independent of the underlying communications protocols; failures of these protocols do not automatically cause the Session to terminate. Sessions terminate based on Client or Server request, or based on inactivity of the Client; a Session has a timeout that allows the Server to free the resources for a Session after a defined time period. The inactivity time interval is negotiated during Session establishment.

A Subscription is the context to exchange data changes and Event Notifications between server and client. A Subscription requires a Session to transport the data to the client. Clients control the rate at which publishing occurs by sending Publish Messages. Subscription lifetime is independent of the Session lifetime and a Subscription has a timeout that gets reset every time data or keep-alive messages get sent to the client.

Multiple Monitored Items can be created in a Subscription. A Monitored Item is used to define the Attribute of a Node that should be monitored for data changes or to define the Event source that should be monitored for Event Notifications. Monitored Items are entities in the Server created by the Client that monitor AddressSpace Nodes and their real-world counterparts. When they detect a data change or an event/alarm occurrence, they generate a Notification that is transferred to the Client by a Subscription. Each Notification contains a sequence number that allows Clients to detect missed Notifications. When there are no Notifications to send within the keep-alive time interval, the Server sends a keep-alive Message that contains the sequence number of the next Notification sent. If a Client fails to receive a Notification after the keep-alive interval has expired, or if it determines that it has missed a Notifications, it can request the Server to resend one or more Notifications.

C. Service Sets

As said before, Client/Server communication is realised by OPC UA Services, which are the interface between OPC UA Clients and Servers. These Services are organized into logical groups called Service Sets.

In the following some of the OPC UA Services defined in the Service Sets, will be presented. The description will be limited only to those services which seems mostly involved in the performance evaluation of OPC UA.

The Read and Write Service Set is used to read and write one or more attributes of one or more Nodes. They allow also reading/writing subsets or single elements of array values. Like most other Services, the Read and Write Services are optimised for bulk read/write operations and not for reading/writing single values.

A different way to access data is the subscription for data changes. This is the preferred method for clients needing cyclic updates of variable value changes. In order to realise this type of data exchange, the Monitored Item Service Set and the Subscription Service Set are used.

The Monitored Item Service Set is used by the Client to create and maintain Monitored Items. 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 to monitor and by defining a filter for the Event. The third type of Monitored Item is used to subscribe for aggregated Values calculated based on current Variable Values in client-defined time intervals. A subscription can contain all three different types of Monitored Items and the Server will deliver notifications of these Monitored Items until the Subscription or the Monitored Items are deleted. As said before, Notifications are generated when certain conditions defined for each Monitored Item occur.

All Monitored Items have common settings: sampling interval, monitor mode, queue size and filter settings. The sampling interval defines the rate the server checks Variable Values for changes or defines the time the aggregate get calculated. The sample rate defined for a Monitored Item may be faster than the publishing rate of the relevant Subscription; for this reason, the Monitored Item may be configured to either queue all Notifications or to queue only the latest Notification for transfer by the Subscription; in this latter case, the queue size is one. Monitored Item Services also allow to set a monitoring mode. The monitoring mode is configured to disable sampling and reporting, to enable sampling only, or to enable both sampling and reporting. When sampling is enabled, the Server samples the item; in addition, each sample is evaluated to determine if a Notification should be generated, and in this case the Notification is queued. If reporting is enabled, the queue is made available to the Subscription for transfer.

The Subscription Service Set is used by the Client to create and maintain Subscriptions. There are two Subscription settings. One is the Publish interval, which defines when the server clears the queues and delivers the Notifications to the client. The other setting is the Publish enabled which defines whether the data gets delivered to the client.

The only two Services used to actually deliver the Notifications to the client are the Publish Service for transferring the Notifications and the Republish Service to get lost Notifications from the server.

It is expected that a client sends a list of Publish Service requests without expecting an immediate response. The server can queue the Publish request until a Notification ready for sending to the client (according to the Publish interval, as said before). 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.

D. Security Model

OPC UA security is concerned with the authentication of Clients and Servers, the authentication of users, the integrity and confidentiality of their communications, and the verifiability of claims of functionality. OPC UA provides a security model, in which security measures can be selected and configured to meet the security needs of a given installation. This model includes security mechanisms and parameters. This framework also defines a minimum set of security Profiles that all OPC UA Servers support, even though they may not be used in all installations.

Application level 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. This means users need to be authenticated only once, when the application Session is established. 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 organizations may also be exchanged during Session establishment. The Server further authenticates the user and authorises subsequent requests to access Objects in the Server. Authorization mechanisms, such as access control lists, are not specified by the OPC UA specification. They are application or system-specific.

OPC UA includes support for security audit trails with traceability between Client and Server audit logs. If a security-related problem is detected at the Server, the associated Client audit log entry can be located and examined. OPC UA also provides the capability for Servers to generate Event Notifications that report auditable Events to Clients capable of processing and logging them. OPC UA defines security audit parameters that can be included in audit log entries and in audit Even Notifications.

OPC UA security complements the security infrastructure provided by most web service capable platforms. Transport level security can be used to encrypt and sign Messages. Encryption and signatures protect against disclosure of information and protect the integrity of Messages. Encryption capabilities are provided by the underlying communications technology used to exchange Messages between OPC UA applications.

III. 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 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 is the most important requirement. 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.

According to what said, performance evaluation seems to be needed in order to verify that requirements in terms of speed, load and complex data structure handling featured by industrial applications, are meet by OPC UA architecture.

Performance should be evaluated taking into account all the aspects and the mechanisms of the OPC UA specification which could influence the behaviour of industrial applications using OPC UA to exchange information. Furthermore the most meaningful parameters must be chosen to achieve significant performance measurements.

The aim of the following subsections is to point out both the main features of the OPC UA specification candidate to influence the relevant performance, and the parameters that seems most suitable to achieve meaningful performance measurements.

A. Security

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

OPC UA is used at different levels of the automation pyramid. 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 act as client and server at the same time; it could be the client collecting data from the server at the lower level and performing special calculations, generating alarms, or performing operations whereby the results are presented to other OPC UA clients (e.g. applications monitoring the state of the production process). At the very top level (i.e. corporate network), an OPC UA client integrated in an ERP system could obtain information about used devices in the plant floor (e.g. working hours) and could create a maintenance request; in addition the corporate network layer could allow remote access via Internet to OPC UA servers in order to perform service or maintenance tasks.

The example scenario just depicted shows that OPC UA can be used at various places and for different applications within the same environment; the security requirements for these applications may also differ in various ways. 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 the requirements could be completely different: performance could be more important than security when data has to be acquired in 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 needs 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 (CA) while opening a secure channel; the other aspect is relevant to the encryption/decryption and the signature of the messages exchanged between OPC UA client and server (e.g. read/write single or set of variables).

In e-commerce environment a waiting time of 5-10 seconds 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 seconds can be a very long time interval for industrial applications, especially for devices located at the field level of the automation pyramid. Sometimes a delay like that above-mentioned is even far long; this happens for example considering applications in chemical or pharmaceutical industries, where very little 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. An operator workplace supervising a special area of a power plant for example 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. 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 acquire useful values of variables and events to properly manage fault or emergency, only during the time period needed to resolve the problem; furthermore any delay in the connection should be avoided.

As known, encryption allows to achieve confidentiality in the data exchange between client and server. 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, considering field level devices, including PLC, Controllers and HMI/SCADA. The same considerations must be extended to the digital signatures present in the OPC UA specification 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.

B. Transport Protocols and Encoding Rules

As said before, OPC UA uses different transport technologies; for embedded systems and UA products used in an automation environment, the preferred transport mechanism is the optimised UA TCP protocol with binary encoding. For enterprise systems, the preferred mechanism may be Web Services using binary or XML encoding. Performance evaluation should take into account the different transport protocols and encoding rules, comparing their impact on the data exchange.

C. Subscription

Subscription is the mechanism able to guarantee that information produced in a cyclic fashion, could be delivered to the client in the same order as they are produced and with the same production period or the same intervals between each pair of subsequent productions.

The previous section pointed out that the subscription is based on several parameters which should be taken into account when performing a performance evaluation, as they may really influence the performance of the client/server data exchange.

Among the parameters to be considered, there are the common settings of the Monitored Items: sampling interval, monitor mode, queue size and filter settings. In particular sampling interval and queue size seems to play an important role in the overall performance, as the sampling interval defines the rate at which the server checks Variable Values for changes and the choice of the queue size is strongly linked to the publishing rate of the Subscription.

As known, the subscription is mainly based on Publish requests sent by the client; the server can queue the Publish request until a Notification is ready for sending to the client. To make sure that all Subscriptions can send a notification at the same time, the client should make sure that there are more outstanding Publish requests than active Subscriptions. The number of outstanding Publish requests a client should maintain is a very critic parameter and could influence the overall performance of the system. For example, additional Publish requests may be required if the latency of the network connection is very high; in the case of a large number of Subscriptions and low network latency, the number of outstanding Publish requests can be reduced.

D. Performance Measurements

Generally industrial applications feature two different kinds of data exchange. The first may be called aperiodic, and occurs when a client (e.g. SCADA) requires the (read and/or write) access to one or more variables at unforeseeable time intervals. The second kind of data exchange is cyclic or periodic and occurs when a client accesses to values of variables according to a periodic signals; an example is a client which needs to access a temperature value produced by a sampling algorithm operating at the sampling period.

In the two different scenarios the single piece of information to be transmitted could be simple (e.g. an integer, a float) or complex (e.g. a data structure made up by several bytes or a large set of variables).

On the basis of what said, the parameters which should be taken into account for meaningful performance measurements will be pointed out in the following.

Considering the first kind of data exchange (the aperiodic one), it seems suitable the choice of round-trips necessary for a typical Read/Write method call depending on the size of Variables included in the bulk operation; see next section for a better definition of round-trips.

Considering the periodic data exchange, the choice of the jitter parameter seems suitable. Considering a particular variable, whose values are periodically produced on the server-side, jitter may be defined as the time interval between the instant at which each values has been produced and the instant at which the client receives that value. On the basis of this definition, it's clear that average jitter for each variable, provides for a measurement of the efficiency of the data exchange, and in particular it points out the capability to deliver to the client each information periodically produced in the exact order of the production and with the same periodicity, without any loss of samples.

IV. Performance Evaluation Results

The previous section highlighted the main features of the OPC UA specification which seem, more than others, able to influence the relevant performance. On the basis of the outcomes of the section, some of the results of a performance evaluation of OPC UA specification carried on by the authors will be presented in the following.

The results here presented will be limited to the security aspects of OPC UA specification outlined in the previous section, i.e. the verification of certificates and the encryption and signature of each message exchanged between OPC UA client and server.

It's very important to point out that OPC UA specification doesn't make mandatory the use of certificates, digital signatures and data encryption; this improves the strategic role of a performance evaluation aimed to highlight the overhead of validation of certificates, encryption/decryption and signature of each message exchanged. Results of the performance evaluation 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.

The following scenarios have been considered during performance evaluation of the OPC UA security: (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 Certification Authority-CA) and (4) use of the Secure Channel with remote validation of the certificates (i.e. operated by a remote hierarchy of CAs).

For all the above-mentioned scenarios (except the first one), the following sub-cases have been considered: (a) no other security option used, (b) use of signature for each message exchanged, and (c) use of signature and data encryption/decryption for each message exchanged.

Combining all the previous scenarios and sub-cases, 10 types of performance evaluation have been considered; they will be indicated in the following using a number (ranging from 1 to 4) and a optional letter (a, b and c). The number refers to one of the previous 4 scenarios and the letter refers to one of the previous 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. As said before, scenario 1 is not featured by any sub-cases, so performance evaluation of type 1 refers to the scenario 1, i.e. data exchange with no security mechanisms.

For each type of performance evaluation, both UA TCP and SOAP transport mechanisms have been considered; in both cases, only binary encoding has been assumed. Furthermore for each type of performance evaluation a couple of OPC UA client and server exchanging data assuming different available bandwidths, has been considered; all the results present in this section refer only to a data exchange between a OPC UA client and server assuming an available bandwidth of 2 Mbps.

Performance evaluation has been realised considering OMNeT++ simulator [5].

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.

Figure 3 compares, in a logarithmic scale, the times needed to activate a session between OPC UA client and server, including the times needed to open the channel and to open the session inside the channel. The figure considers only the four scenarios, without the sub-cases, as they have no influence on the activation of a session. The figure considers the results achieved considering both the two transport mechanisms (UA TCP and SOAP). Table 1 details the times needed to activate a session.

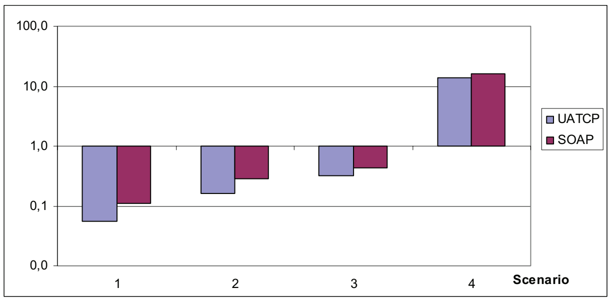

Fig. 3. Times needed to activate a session.

Table 1: 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

Both figure and table point out the influence of the transport mechanism on the times needed to activate a (secure or not) session. In the case of secure session with the use of remote certification authorities, the times needed to activate a session are very huge and use of UA TCP leads to a very little save in time (about 1 second).

Performance evaluation has been carried on also to investigate the influence of signature and encryption on the round-trip times; round-trip time is defined as the total response time between the instant at which a request to read one or a set variables is issued by a client and the instant at which the relevant values are delivered to the client.

Figure 4 compares the round-trip times achieved considering the performance evaluation types 1 and 2.c and the two different transport mechanisms; each curve shown in Figure 4 is obtained normalizing the values of the round-trip time to those achieved considering scenario 1 (no security) and UA TCP. So, curve labeled with "2.c UA TCP" refers to the round-trip values concerning scenario 2.c and UA TCP, normalized to the values achieved considering scenario 1 and UA TCP; curve "2.c SOAP" refers to the round-trip values considering scenario 2.c and SOAP normalized to the values achieved considering scenario 1 and UATCP. Finally curve "1 SOAP" refers to the round-trip times relevant to scenario 1 and SOAP normalized to the scenario 1 and UA TCP.

Evaluation of the round-trip times has been achieved considering different sizes (in bytes) of the set of variables read from the server, shown in the abscissa of Figure 4.

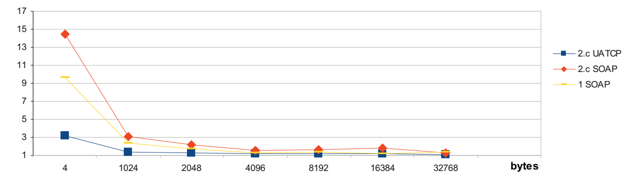

Fig. 4. Round-trip delay considering scenarios 1 and 2.c and both UA TCP and SOAP

Figures 5 and 6 compare the round-trip times considering the scenarios (1, 3.c) and (1, 4.c), respectively. Again the round-trip values are normalized as explained before.

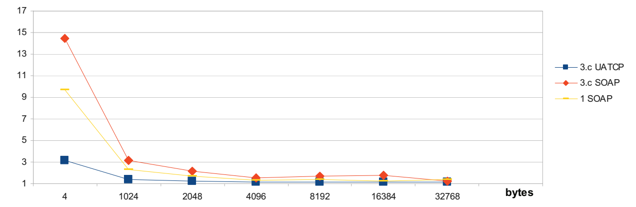

Fig. 5. Round-trip delay considering scenarios 1 and 3.c and both UA TCP and SOAP

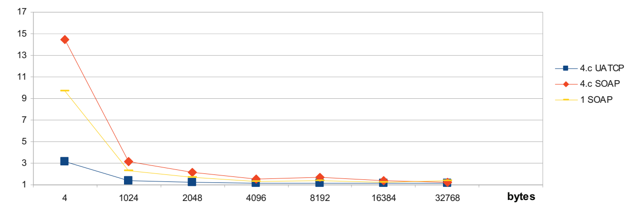

Fig. 6. Round-trip delay considering scenarios 1 and 4.c and both UA TCP and SOAP

Figures 4, 5 and 6 point out that use of UA TCP leads to lower round-trip times, also in presence of the security mechanisms. This mainly occurs with small size of data exchanged; when the size of variables exchanged increases, the performance of the UA TCP and SOAP tends to converge, also in presence of security mechanisms.

The last set of measures about security here shown, are relevant to the interest to investigate the influence on the overall performance of the different configurations it's possible to choose inside a secure channel; these configurations are relevant 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 3 different scenarios (2, 3 and 4) which include the 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.

Figure 7 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 normalized to those achieved considering scenario 1 (no security) and UA TCP. As can be seen from the figure, the influence of the signature and encryption mechanisms is very huge for small size of variables exchanged; performances of the different scenarios converge when variables increase in size.

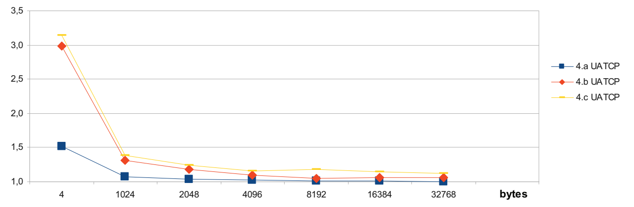

Fig. 7. Round-trip delay considering scenarios 4.a, 4.b and 4.c and UA TCP

V. Conclusion

The paper has presented the OPC UA specification, pointing out its widespread use in the current industrial informatics; some features of the specification which may produce an impact on the overall performance of the client/server data exchange have been pointed out. The paper has then proposed some measurement parameters to be considered in the performance evaluation, which seem suitable to point out how performance of OPC UA – based data exchange may be optimized tuning certain parameters of the specification. Finally, some of the main results achieved during performance evaluation have been presented and discussed, focusing on those relevant to the OPC UA security mechanisms.

References

[1] Braune A, Henning S, Hegler S (2008) Evaluation of OPC UA Secure Communication in Web Browser Applications. In Proc.IEEE International Conference on Industrial Informatics (INDIN 2008), Daejeon, Korea, pp.1660-1665.
[2] Post O, Deppala J, Koivisto H (2009) The Performance of OPC UA Security Model at Field Device Level. In Proc. ICINCO , pp 337–341.
[3] Mahnke W, Leitner S-H, Damm M (2009) OPC Unified Architecture. Springer, ISBN 978-3-540-68898-3.
[4] OPC Foundation (2009) OPC UA Specification: Part 1 – 13.
[5] www.omnetpp.org