Communications and Network, **2011, 3, 39-49**
doi:10.4236/cn.2011.31006 Published Online February 2011 (http://www.SciRP.org/journal/cn)

Scientific Research

Embedding Authentication & Authorization in Discovery Protocols for Standard Based Publish/Subscribe Middleware: A Performance Evaluation

Fabrizio Ronci, Marco Listanti
DIET Department, SAPIENZA – University of Rome, Rome, Italy
E-mail: *ronci@net.infocom.uniroma1.it, marco@infocom.uniroma1.it*
Received October 27, 2010; *revised December* 22, 2010; *accepted February* 15, 2011

Abstract

Standard based Pub/Sub middleware, such as OMG Data Distribution Service (DDS), could assume a key role in supporting computer communications requiring continuous state information updating, deterministic deadline to data delivering and real time information adjourning. This kind of capability could be well exploited by Peer-To-Peer (P2P) systems, Internet-wide as long as private ones, like in Public Safety or Civil Protection Communication Systems; but Pub/Sub specifications, and DDS/RTPS (Real Time Publish Subscribe) as well, usually do not provide Authentication & Authorization (AA) mechanisms. In the present work two important novelties are assessed: a possible scheme to implement AA in DDS/RTPS networks and a time performance evaluation study about embedded Authentication in RTPS.

Keywords: Middleware, Publish/Subscribe, Data Distribution Service, Authentication, Authorization

1. Introduction and Motivation

Pervasive computing applications have been increasingly relying on event based communications during the very last years; concurrently there has been more and more interest in developing Publish/Subscribe schemes able to implement and connote such new paradigms.

Much of the event based interaction style potential is in the total decoupling between information producers and consumers along several dimensions: time, space, synchronization. This kind of decoupling is mostly often implemented by a Pub/Sub middleware [1], where producers, or publishers, and consumers, or subscribers, commit collocated agents or brokers, belonging to a network laid on the actual network of communicating entities, to exchanging events and messages.

A great number of industrial implementations and academic prototypes have been released since it was recognized that networking equipment and protocols technologies had reached enough power and reliability to make the switch from Request/Reply architectures to Publish/Subscribe ones convenient. Nevertheless such a restless growing in conceiving, designing and implementing Pub/Sub middleware solutions, very diverse in scope, field of application, programming model and extendibility, has possibly prevented until now the large scale adoption of Pub/Sub communication services directly at the Internet end-user premises.

It appears likely that future massive usage of Pub/Sub systems in Internet-wide applications will happen if event based communications can be demonstrated suitable for highly popular web utilities: every web utility application requires in fact a continuous information updating on peers' state. Some recent research [2] has proposed that Pub/Sub schemes would perform better than traditional interactional ones in supporting this kind of applications because of their capability to offer deterministic deadline to data delivering and real time information adjourning to the overall set of participating nodes.

A middleware infrastructure is a shared belief to be at the heart of every Pub/Sub communication system; and usually such an infrastructure is intended to be composed of multiple distributed agents. Agent based infrastructures:

- facilitate communications among entities, *i.e.* applications, with rich and eventually different semantics;
- enable transparency in respect of underlying technologies and dynamicity in respect of time and throughput application demands;

- give nodes, as a collection of application entities, a certain degree of autonomy, posing the basis for autonomic behaviour.

These opportunities gave researchers and companies space to customize the agent based Pub/Sub paradigm in several ways, case by case emphasizing specific aspects to be obtained: high performance transfer and processing of streams, context awareness and situation detection, transparent ad hoc communication, as well as autonomic features [3].

Of primary concern to these real time custom implementations was and is the efficient distribution of data with minimal overhead and the ability to control QoS properties. Distributed shared memory is a classic model that provides data-centric exchanges. However, this model was and is difficult to implement efficiently over the Internet. Therefore, another model, the Data Centric Publish Subscribe (DCPS) model, has become, since the early 2000s, popular in many real time applications. While there have been several commercial and in-house developments providing this type of facility, to date, there had been no general purpose data distribution standards, until Object Management Group (OMG) published the Data Distribution Service (DDS) specification [4].

In the present work two important novelties are assessed: a possible scheme to implement AA in DDS/RTPS networks, overcoming some difficulties in the standard; and a time performance evaluation study about embedded Authentication in RTPS, which takes on its basis from the Discovery procedure, and hence is the first, at the best of our knowledge, simulation effort about network formation in standard Pub/Sub middleware.

Actually, in distributed agents theory it is very common to consider a specialized component, within the node architecture, devoted to encompass any other one and to offer general services to applications, receiving information from underlying layers: in DDS/RTPS, on the contrary, all components, related eventually to different information domains, are independent and not bound to a fixed hierarchy. That appears as a tough obstacle to implement AA because it is not defined how to manage the specification in order to identify where to pose such services. The present document shows a feasible approach to come through these limitations, in absence of standardized procedures.

Meanwhile, in most of research papers on networking, networks are supposed to be already formed when proposed protocols or solutions are posed under evaluation. As known, network formation is in any case a key feature and lower layer standards, 802.x as well as MANET routing for example, spend great attention on detailing them. Obviously it is not possible to consider a task for a Pub/Sub middleware addressing such a feature, because middleware is placed at transport and application layers, but it is straightforward and in some way mandatory for middleware to realize network formation at its own level through well designed discovery protocols: if, when and how to relate these protocols with lower layers network formation is an interesting, open and possibly fruitful field of research and the present work, simulating embedded AA in DDS/RTPS Discovery Protocols, besides exposing the first network formation simulation for this standard based Pub/Sub middleware, is a first implementation scheme able to allocate cross layer interactions among PHY/MAC network access coordination methods, routing construction algorithms and middleware discovery functions.

2. Data Distribution Service and Real Time Publish Subscribe

2.1. DDS – Data Distribution Service

Object Management Group Data Distribution Service (OMG DDS) utilizes the Pub/Sub paradigm to achieve highly efficient information dissemination between multiple publishers and subscribers that share interest in so-called Topics. This specification also includes a QoS framework allowing the middleware to match requested and offered (RxO) Quality of Service parameters, on the basis of attributes like reliability, ordering and urgency.

OMG DDS is a Publish/Subscribe middleware that operates upon a Global Data Space; in perspective, in a security and service isolation context, each Global Data Space represents a single administrative domain.

Central concept in OMG DDS is that of Topic: this is a data structure, described in terms of an unique name – in the Global Data Space – and a type, but accompanied by QoS attributes.

Topics are typically objects and their instances are univocally identified by the middleware in a single data space by means of a key, taken from one of the data type elements. Matching between publishing and subscribing participants in a data domain is carried out at all data structure components: name, type and QoS policy.

OMG DDS offers means to cope with challenging communication systems requirements as well as with pervasive computing data users requirements.

Selection of OMG DDS specification for our work originates from considering integration and interoperability requirements: data centricity, in respect of actors and technologies, and transparency, up to layers as higher as possible. OMG DDS offers such features as it allows compatibility among technologies up to the trans-

port layer, and methods directly at data level, through an interoperability protocol.

2.2. RTPS – Real Time Publish Subscribe Interoperability Wire Protocol

As the DDS specification defines an Application Level Interface and a behaviour supporting Data-Centric Publish-Subscribe (DCPS), *i.e.* the lower service component, in real-time systems, it addresses specifically application portability. On the other hand it does not address, on its own, either portability among schemes implementing the messages exchange requested by DDS entities or among transport protocols claimed to move such messages, like typically TCP/UDP/IP.

One of the DDS specification prescriptions is the presence of a built-in discovery service that allows publishers to dynamically, continuously and without the need to contact any name servers discover the existence of subscribers and vice-versa.

With the increasing adoption of DDS, it has been appreciated desirable to define a standard wire protocol, allowing diverse DDS implementations to interoperate: such a protocol to be capable of taking advantage of the QoS settings configurable by DDS, of optimizing underlying transport capabilities, and of exploiting the multicast, best-effort, and connectionless OMG DDS communications.

Real Time Publish Subscribe (RTPS) wire protocol is that protocol: as a direct result, a close synergy exists between OMG DDS and RTPS, both in terms of the underlying behavioural architecture and the features of RTPS. The RTPS protocol is designed to be able to run over multicast and connectionless best-effort transports such as UDP/IP.

RTPS protocol is specified by OMG in terms of a Platform Independent Model (PIM) and a set of Platform Specific Models (PSMs) [5]. The RTPS PIM contains four modules:

- *Structure Module*, defining the communication endpoints;
- *Messages Module*, defining the set of messages that endpoints exchange;
- *Behaviour Module*, defining sets of endpoints legal interactions and state variation;
- *Discovery Module*, defining how RTPS entities, namely *Participants* and *Endpoints*, are automatically discovered and configured to interact with their DDS counterparts.

Despite their names, RTPS Modules are not interacting parts, but rather sections in the PIM specification each describing a general aspect necessary to implement the middleware. As usual in standards, these aspects are to be intended as minimum requirements, leaving to single implementations some degrees of freedom on how to realize them.

In other words, the *Structure Module* defines which protocol actors have to be present, the *Messages Module* which are the grammatical symbols these actors have to use and how they have to construct them and the *Behavior Module* defines the legal grammar, *i.e.* a minimum set of rules, and semantics of the different conversations, among above mentioned actors and made of above mentioned symbols. The *Discovery Module* defines how entities are automatically discovered and configured. In the PIM, the messages are defined only in terms of their semantic content. This PIM can then be mapped to various Platform Specific Models (PSMs), plain UDP or CORBA events for example.

In the present work, all the focus is on the latter *Module*, but it appears useful to recall here some *Structure Module* items:

- all RTPS entities are associated with an RTPS *Domain*, which represents a separate communication plane that contains a set of *Participants*;
- a *Participant* contains local *Endpoints*;
- there are two kinds of *Endpoints*: *Readers* and *Writers*, which are the actors that communicate information by sending RTPS messages;
- RTPS Entities are in one-to-one correspondence with the DDS Entities that are the reason for the communication to occur: in particular each RTPS *Participant* is in one-to-one correspondence with a single *DDS Domain-Participant* (**Figure 1**).

As explained in the following, the very last clause is relevant to designing an Authentication & Authorization scheme for RTPS Entities.

2.3. RTPS Discovery

The *Discovery Module* describes the protocol(s) that enables *Participants* to obtain a complete picture of all remote *Participants*, *Readers* and *Writers* in the *Domain* and configure the local *Writers* to communicate with the remote *Readers* and the local *Readers* to communicate with the remote *Writers*.

The unique needs of Discovery make it unlikely that a single architecture or protocol can fulfill the extremely variable scalability, performance, and "embeddability" needs of the various heterogeneous networks where DDS/RTPS will be deployed. Henceforth, it makes sense to introduce several discovery mechanisms ranging from the simple and efficient (but not very scalable) to the more complex hierarchical (but more scalable too) one.

RTPS specification splits up the Discovery protocol into two independent protocols. A Participant Discovery

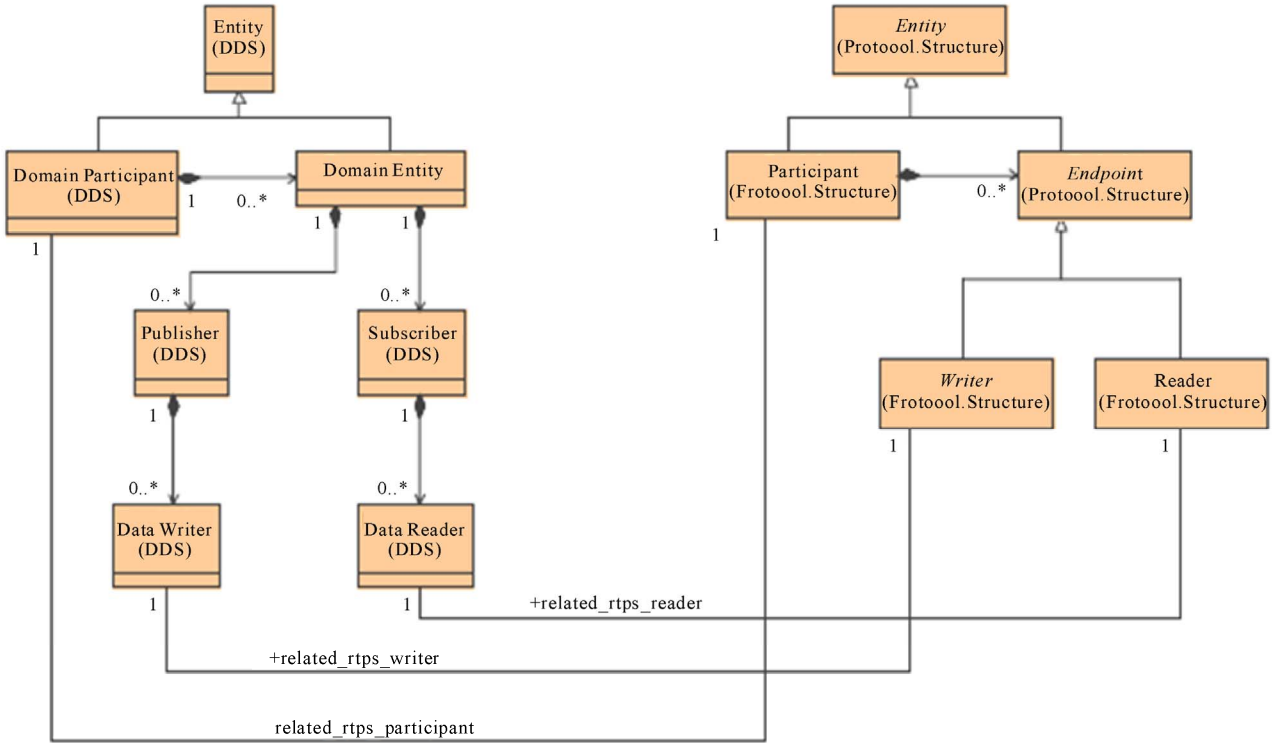


Figure 1. UML description of 1-to-1 correspondences between DDS and RTPS actors. (source:[5]).

Protocol (PDP) specifies how *Participants* discover each other in the network. Once two *Participants* have discovered each other, they exchange information on the *Endpoints* they contain using an Endpoint Discovery Protocol (EDP). Both protocols rely on pre-defined RTPS *built-in Writer* and *Reader Endpoints*.

For the purpose of interoperability, all RTPS implementations must provide at least the following discovery protocols, which are specified:

- Simple Participant Discovery Protocol (SPDP);
- Simple Endpoint Discovery Protocol (SEDP).

To the aim of this work it suffices considering only PDPs, firstly the Simple one and then an Extended – not standard – version, with embedded AA features.

The RTPS SPDP uses a simple approach to announce and detect the presence of *Participants* in a *Domain*. For each *Participant*, the SPDP creates two RTPS *built-in Endpoints*: the *SPDPbuiltinParticipantWriter* and the *SPDPbuiltinParticipantReader*. As these *built-in Endpoints* act as regular *Writer* and *Reader Endpoints*, they use the regular RTPS protocol defined in the *Behaviour Module*, by means of the *HistoryCache*, a collection of data-objects which is the fundamental interfacing part between DDS and RTPS. In other words, both the RTPS Entities and their related DDS Entities are able to invoke the operations on their associated, and common, *HistoryCache*.

The *HistoryCache* of the *SPDPbuiltinParticipantWriter* contains a single data-object of type *SPDPdiscoveredParticipantData*. The *SPDPbuiltinParticipantWriter* periodically sends this data-object to a preconfigured list of *Locators* to announce the *Participant* presence on the network. The pre-configured list of *Locators* may include both unicast and multicast *Locators*. By sending the *SPDPdiscoveredParticipantData* periodically, *Participants* can join the network in any order. The *SPDPdiscoveredParticipantData* defines the data exchanged as part of the SPDP and contains a list of attributes, derived from the relevant *Participant*.

The *HistoryCache* of the *SPDPbuiltinParticipantReader* contains information on all active discovered *Participants*; the key used to identify each data-object corresponds to the *Participant* Globally Unique IDentifier (GUID). Each time newer information on a *Participant* is received by the *SPDPbuiltinParticipantReader*, the SPDP examines the *HistoryCache* looking for an entry with a key that matches the *Participant* GUID. If an entry with a matching key is not there, a new entry is added keyed by the GUID of that *Participant*.

3. Authentication & Authorization in RTPS

The RTPS protocol does not provide any Authentication mechanism, nor the DDS specification has standardized

yet security mechanisms. Because of this a scheme is going to be suggested in order to provide a Pub/Sub middleware with a possible form of Authentication model. The intended way is to review and adapt the classic challenge and response model to the Pub/Sub environment. End-users are here not willing to query or access a particular server or other network node which they will initiate a communication with, but they are belonging to one or more domains, which they will be publisher and/or subscriber of data within.

RTPS Discovery represents a process by which, within a single *Domain*, each node is made aware of the presence of other nodes. More specifically, the SPDP acts the mutual discovery among the RTPS *Participants* in that *Domain*. Most of private computer or radio networks implement access security giving user devices identity credentials, valid for a first phase of Authentication, and then basing Authorization on Authentication: as indicated above, RTPS protocol does not recognize to any specific *Participant* a guiding role in Authentication protocol, nor, it is the same due to the one-to-one correspondence, recognize to any specific *Domain* the value of Global Space for identity procedures and data.

This could be an intrinsic limitation to embedding Authentication procedures in DDS/RTPS middleware.

3.1. Proposed Scheme

In the present work a feasible solution is set to overcome such a difficulty posed by the specification: the proposed answer is to bring together and at an upper, applicative, level, the functions of Discovery, Authentication providing and Authorization providing, in an unique *Domain*, which could be identified as **Network Presence** or **Participation**.

Starting from such a DDS Domain, RTPS modules, one Writer and one Reader, pre-configured for Discovery (built-in Endpoints) are generated: these modules embody the Authentication function and are driven by the Authentication application present in every network node.

At the same time, within the same Domain, are created a number of Partitions as many as the number of communication applications supported by the network: in every Partition one Writer and one Reader are generated too, in order to provide separated Authorization function likewise relying on Discovery procedure, one for each application the single node is interested to access to.

Within every application, hence at the applicative layer, this approach allows modules to be exchanging messages with the Network Presence, to trigger the relevant discovery and hence to gain authorization to participate, even through controlled access schemes, Authorization tokens and/or application-specific encryption keys different from that used in Authentication. Messages exchange between Network Presence and other applications and/or between authorization applicative modules and application specific RTPS Discovery Partitions in each node is behind the scene and is beyond the scope of this article.

In **Figure 2** the proposed scheme is depicted: Node 1 does participate to all the three communication applications, while Node 2 does not participate to Application C neither Node 3 to Application B. Neverthless, in Node 2 and Node 3 the Network Presence DDS Domain Participant and RTPS Participant are activated to provide the Authentication function; on the contrary, in DDS, the Domain Participant and the Network Presence Partitions related to the inactive applications are silent, as well as the RTPS Application module and the Authorization Endpoint bounded to those applications.

Remaining on Authentication, in Network Presence RTPS Discovery phase, each *Participant* normally transmits its advertising information and receives those from others. Advertising information, in particular, are sent to a pre-configured list of *Locators*: namely a *Locator* is a triple <Transport Protocol, IP Address, Port> in the RTPS specification PIM and a set of multicast addresses, to which *Participants* have to bind, plus UDP ports in the PSM (the only PSM defined in the specification relies on UDP and that was adopted for our work). On receiving this information, each *Participant* will compose its list of discovered *Participants*, within the Network Presence *Domain*.

To embed Authentication it is needed to assess two assumptions, that may make sense in the case of, for example, Public Safety Communications [6]:

1) *Participants* could share a secret key, even if this is not mandatory;

2) node software components and hardware devices are not tampered or stolen.

Based on considerations above and within the individuated framework, each RTPS Network Presence *Participant* will issue in Discovery messages, which do not require acknowledgement or response, besides usual records in *SPDPdiscoveredParticipantData* like its unique identifier (GUID), both a Nonce, to be varied at every issue (a counter or a timestamp, for instance), and a signature, given by the encryption of a function of GUID and Nonce with the shared secret key.

At the receiver side, recipient *Participant* will compare the message signature with the one it can calculate on its own and will decide the insertion of the issuer in the list of discovered *Participants*. This way each RTPS Network Presence *Participant* will own a list of other *Participants* which all have shared the same secret key,

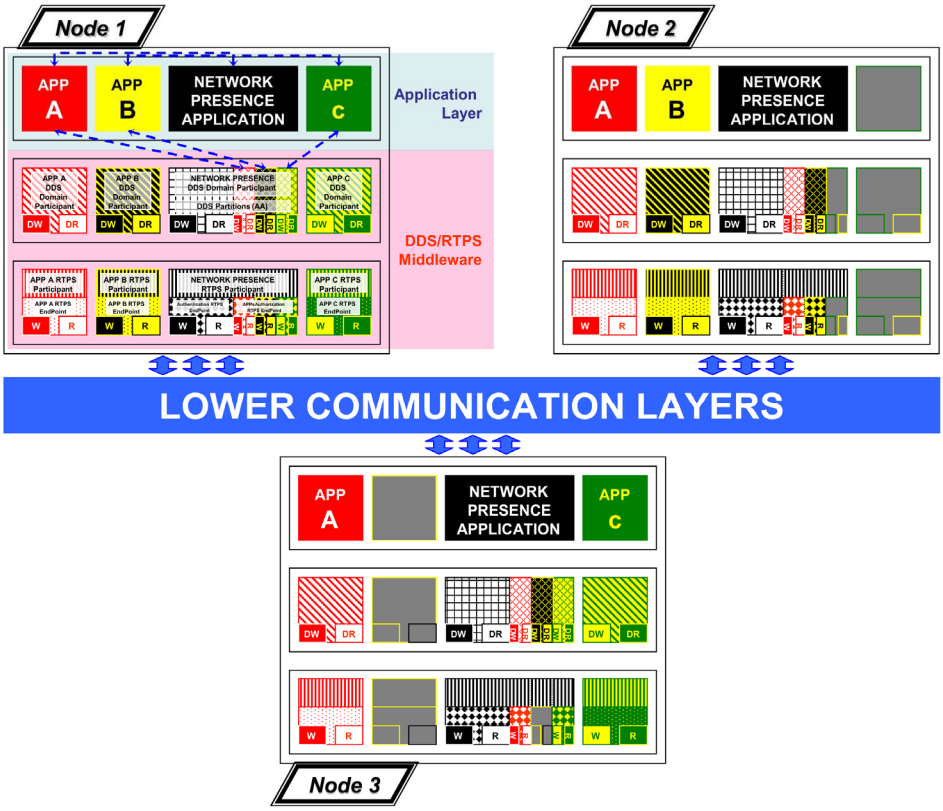


Figure 2. The proposed embedding AA scheme in DDS/RTPS Middleware.

as a mean of Authentication.

It should be noted that the present work is neither bound to a specific authentication scheme nor to the use of a specific type of key: the symmetric scheme, with implicitly block ciphering and a shared, pre-distributed, secret key, has been here adopted for simplicity and without any loss of generality. Different schemes, like asymmetric one for example, with the same approach here proposed could be used for Authentication (*i.e.* in the Network Presence RTPS Discovery), as well as in specific-application authorizing modules (*i.e.* in the specific-application Network Presence Partition RTPS Discovery) could be implemented schemes different from that in Authentication and/or different from that in any other application.

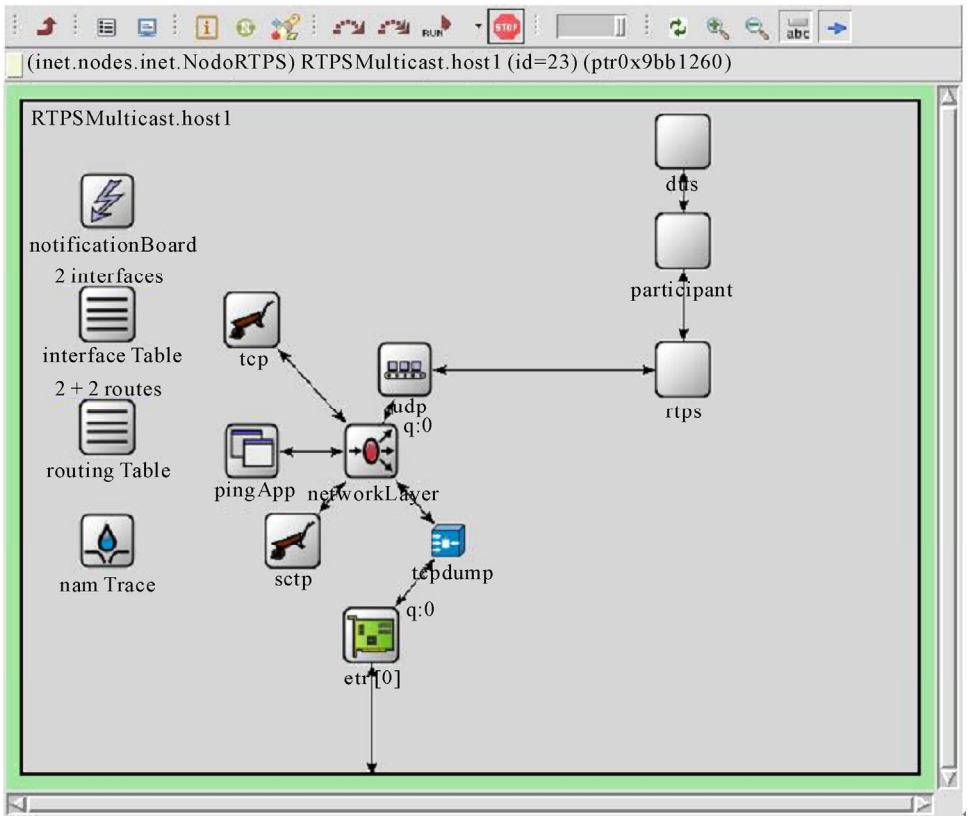
4. Modeling and Simulation

To assess the performance level of the proposed AA scheme in DDS/RTPS middleware, a model has been developed and simulations have been carried out by means of OMNET++ simulator. By the way, as there was simulative development of RTPS *Structure* and *Behaviour Modules*, at the best of our knowledge, this has been the first simulation effort of RTPS *Discovery Module* and hence of network formation in standard Pub/Sub middleware.

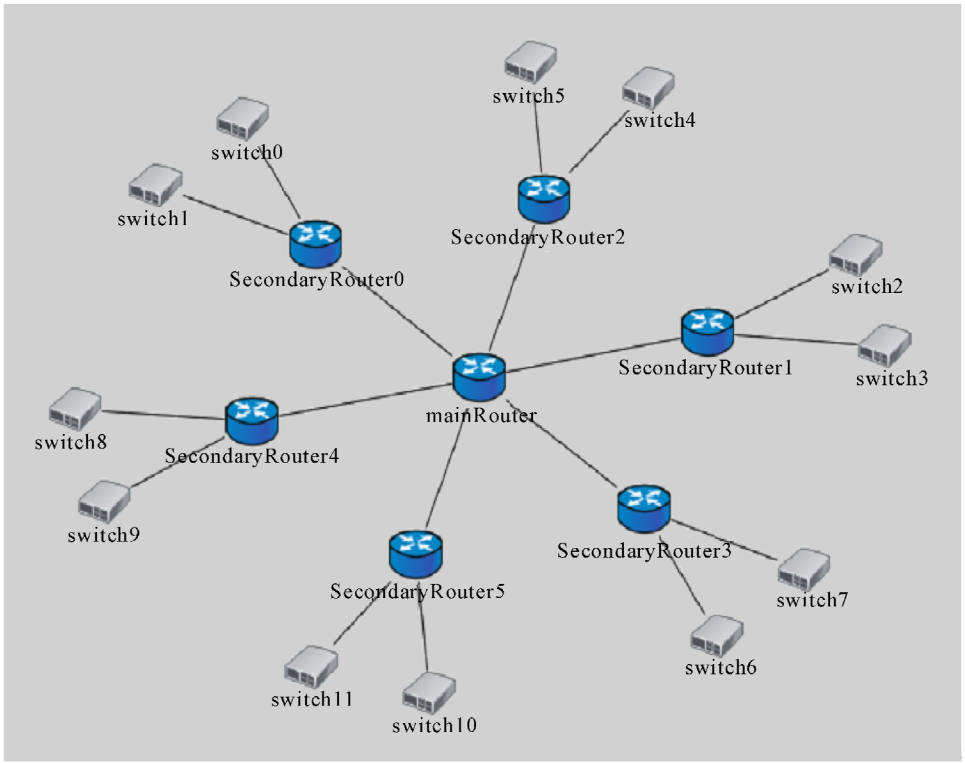
The model is formed by a number of simple and compound OMNET++ modules, derived from the INET OMNET++ Framework. In particular the INET StandardHost module has been extended to become the nodeRTPS (**Figure 3(a)**), a compound module containing the whole protocol stack UDP/IP unchanged from the INET Framework, a semi-applicative level, called DDS module, simulating the application and the DDS behaviours, and a chain of a compound module and a simple module, called respectively Participant module and RTPS module, together simulating the RTPS Discovery behaviour.

OMNET++ simple module RTPS is a buffer component, which has to bind the *Participant* to a specific UDP port and to process and dispatch Discovery messages from UDP to the Participant module and vice- versa.

OMNET++ compound module Participant is composed of two other compound modules, built_in Writer and built_inReader, which dialogue both with DDS module and RTPS module. In each of these compound modules, full functionalities specified by the RTPS protocol are implemented by means of simple modules: a HistoryCache (Publisher side) module, a MessageReceiver



(a)



(b)

Figure 3. (a) NodeRTPS OMNET++ compound module; (b) Fixed skeleton simulation network.

module and a Writer module for the built_inWriter, and a HistoryCache (Subscriber side) module, a MessageReceiver module and a Reader module for the built_in-Reader. Specified RTPS protocol operations syntax and semantics have been both totally respected.

Writer and Reader modules operate the Network Presence Discovery, *i.e.* the Authentication, by means of a shared secret key and the generation of Nonces.

The model is completed by two simple inactive modules, acting as data structure: CacheChange, that permits HistoryCache to maintain the list of discovered authenticated RTPS *Participants*, and a Counter, which serves as a collector for all the information and statistics of a simulation run.

Simulation runs have been carried out upon a fixed skeleton network composed of a central router and six secondary routers, star connected and each provided with two switches (**Figure 3(b)**); nodes, implemented by a number varying from 2 to 150 of nodeRTPS modules, are attached randomly to a different switch at every run. All links are ethernet connections with 100 Mbps data rate and full duplex configuration. Propagation delays are variable at every run, uniformly distributed between 0.5 and 1 ms for secondary-central routers links and between 0.005 and 0.5 ms for secondary routers-switch links: respective BERs are 10^{-8} and 10^{-6}.

Embedding AA in DDS/RTPS Discovery Protocol allows Pub/Sub paradigm to be extended to security functions.

This fact means that key features like Authentication could be realized in a distributed manner, rather than in the common Request/Reply architecture: as known this architecture relies on one or more servers, causing possible critical points of failure and requiring an accurate design to cope with disaster recovery. Furthermore, lower layers technologies play a key role in dimensioning network wideness and resources and very often the choice of reference is SSL/TCP for covert messaging between client and server, leading to a normally huge transport overhead.

The distributed approach described in the present work exhibits some improvements in respect of a centralized one:

- all the nodes can virtually be of the same limited computational power and memory and there is no need of servers provided with quick reply capabilities, huge memory demands and complex database query engines;
- there are no points of failure, because all the knowledge about authenticated nodes is distributed and shared among nodes and, even if one of them crashed, RTPS Discovery resilience (*i.e.* messages periodic repetition) is able to recover a complete and correct network state in a brief lapse of time;
- middlewares, and DDS/RTPS as well, are suited and in general designed to virtualize lower layer characteristics to applications: embedding AA is in such a case a no-cost, no-pain add-on to all other advantages in using a middleware;
- the use of a simple and light transport protocol like UDP is the rule, because of the really minimal requirements in the standard: a multicast addressing capable, connectionless and best-effort transport protocol.

Within this context a direct choice for evaluation is to consider time performances. In fact key length seems to play a very limited role: even with the longest keys for asymmetric schemes RTPS Discovery messages can be easily maintained below the UDP fragmentation limit and encryption/decryption functions, in particular if committed to dedicated circuits, typically do not import significant delay.

As for the reasons explained before, a direct comparison between centralized and distributed AA schemes should not give relevant remarks. Meanwhile it should make little sense to implement Request/Reply AA architecture using a distributed facility like a Pub/Sub middleware and then to compare its performances with that proposed in the present work, where embedding AA in Discovery is a straight way to obtain at the same time network formation and mutual security. In any case it is possible to conduct a qualitative confrontation:

- in distributed embedded AA, delays are mainly caused by collisions and congestion, due usually to number of nodes, link rates and link lengths;
- in centralized AA, a couple, or more, of two-way messages exchanges has to be considered: the first when a node authenticates on the server(s), the second when it requests authorization to access a service provided by another node;
- in distributed embedded AA, authenticated nodes information messages exchange tends to form clouds in local neighborhoods, leaving less charged the core network;
- in centralized AA, a number of valuable resources are deemed to be guaranteed all over the network lifetime.

A baseline simulation, 50 runs for each number of nodes from 2 to 150, has been carried out without packet losses and with nodes accessing the network all at the same instant, T = 0. For every node the time instant when he was made aware of all other ones, *i.e.* constructing a complete list of discovered *Participants*, is collected and averaged with the others. Then an average along different runs was calculated. Results are in **Figure 4**. Such a completion time of Discovery, and hence of Authentica

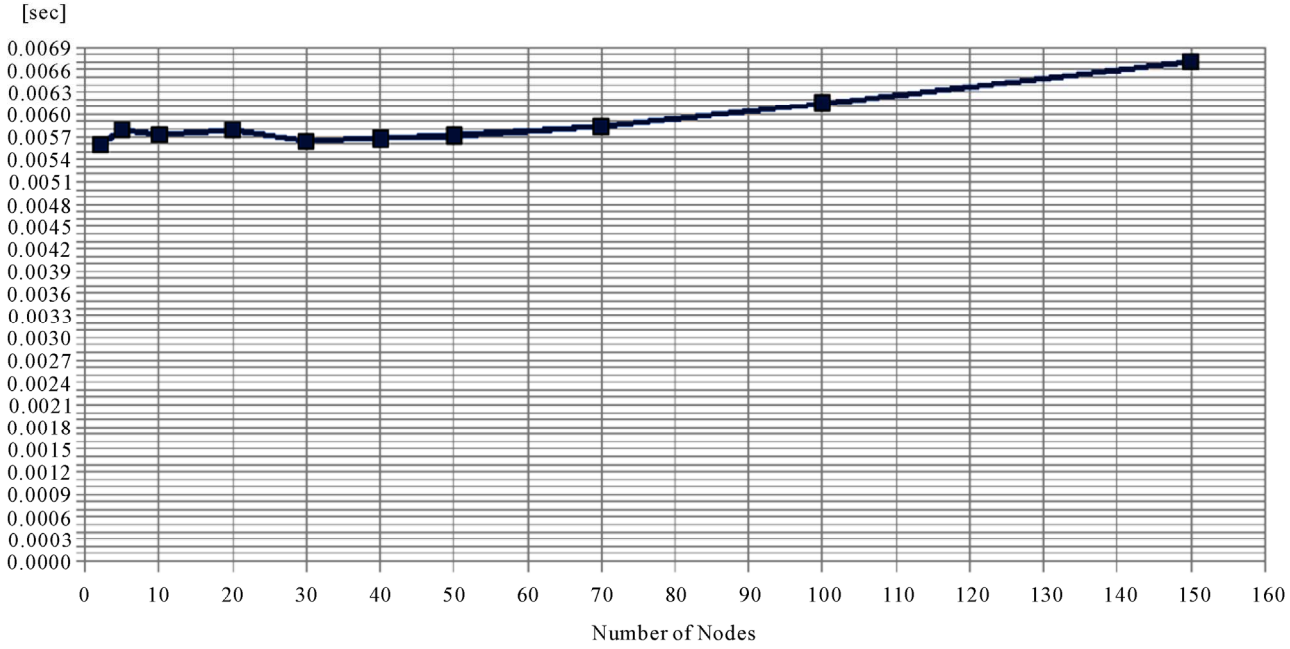


Figure 4. Baseline average mutual authentication completion time.

tion, is quite limited, due mainly to propagation delays and smoothly rising with a number of nodes increase.

A case of interest is when nodes access the network at different instants. Different simulation runs were then fulfilled also varying the time instant when a single node decides to join the network: such an instant has been considered variable and uniformly distributed between T = 0 and T = 0.5; 1; 5; 10; 20; 30 seconds. RTPS protocol states that Discovery messages have to be issued at regular intervals, which individuate a Re-send Interval. In simulation runs Re-send Interval has been made varying among the values 0.5; 1; 5; 10; 20; 30 seconds too.

Extended PDP behaviour has been then compared with a solicited version, where besides the periodic messages issues, each *Participant* is forced to send its list of discovered correspondants every time it had received an advertisement about one or more of them hitherto undiscovered.

For every node and along all the above parameters variations, by means of the Counter module, the time interval between the instant the *Participant* joined the network and the time instant when he was made aware of all other ones has been collected.

In **Figures from 5(a) to 5(d)** are reported, in the case nodes pop up in the network within 30 s since the simulation beginning, respectively the average time of complete authentication for the extended and the extended solicited PDP and the average number of messages issued in the network, in the claimed gamut of Re-send Interval and along 50 simulation runs for each number of nodes.

5. Conclusions

A possible scheme to implement AA in DDS/RTPS networks, overcoming some difficulties in the standard, has been presented, along with a time performance evaluation study about embedded Authentication in RTPS.

Figure 5(a) shows that embedding AA in RTPS Discovery Protocol, when the number of nodes is below 100 and nodes join the network within 30 s, exhibits an average mutual authentication completion time which has a fixed component given by the Re-send Interval and a variable one that increases less than linearly with the Re-send Interval and is independent from the number of nodes; above 100 nodes, the variable component appears to be dependent also from the number of nodes, evidently because of more collisions and congestion.

Figure 5(b) shows that embedding AA and soliciting newer information exchange in RTPS Discovery Protocol can lead to a time performance, interestingly independent from number of nodes and Re-send Interval; below the maximum number of nodes considered collisions do not prevent to maintain overall performance well below the Re-send Interval.

Figures 5(c) and 5(d) depict the cost, in terms of number of exchanged messages, to obtain the performances in the preceding figures. It appears clear how much wasteful is to consider, in the unsolicited extended PDP, a very small Re-send Interval of 0.5 s in respect to a middle one, 10 s for example, even above a limited number of nodes; meanwhile it is appreciable a one-order of

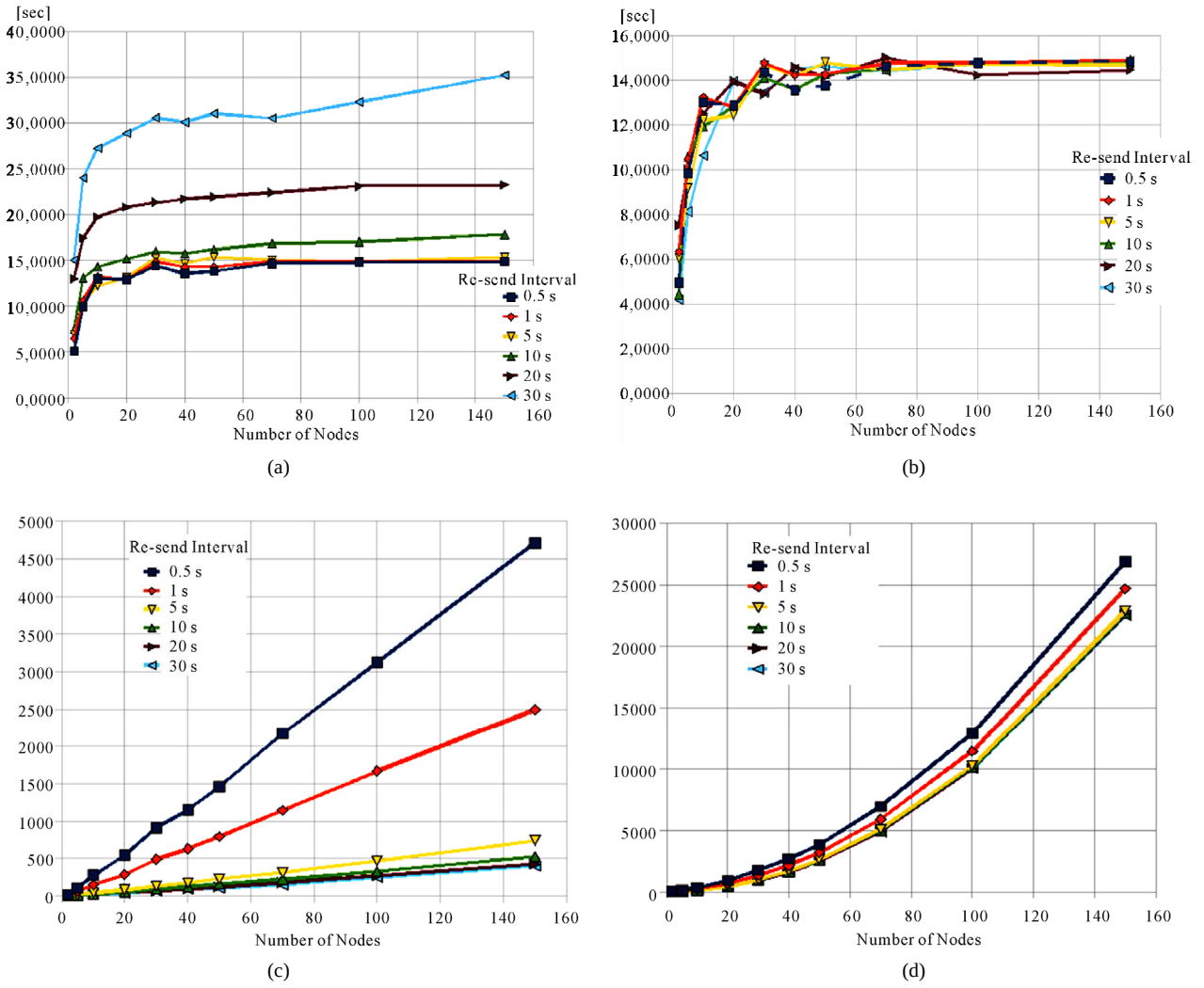


Figure 5. (a) Average mutual authentication (extended PDP) completion time - nodes joining in 30 s; (b) Average mutual authentication (extended solicited PDP) completion time - nodes joining in 30 s; (c) Average exchanged messages number at completion time for extended PDP; (d) Average exchanged messages number at completion time for extended solicited PDP.

magnitude increase in messages exchange when the solicited extended PDP is used, independently from the Re-send Interval.

In conclusion, to obtain low average mutual Authentication completion time within a *Domain*, either – with the extended PDP behaviour – it appears useful to adopt mid Re-send Interval, allowing a linear increase of network traffic, or – with the solicited version – it is feasible to slow down the Re-send pace but expecting local raising in network traffic due to possible nodes access rate peaks. Lastly it is worth noting that, with less than 150 nodes joining the network, there have not been significant decays in Authentication completion time, caused eventually by congestion and collisions.

Simulations showed that exploiting the RTPS Discovery facility, already described in the standard, allows nodes joining the network to be authenticated by peers and enables successive authorizing procedures for each different communication domain, in very limited time intervals, generally well scalable in respect of possible network traffic increases due to greater signalling message exchange.

6. Acknowledgements

This work was supported in part by MIUR-FIRB INtegrated SYstems for EMErgency (INSYEME) under Grant RBIP063BPH.

Experiment implementation has been realized upon OMNeT++ Network Simulation Framework.

7. References

[1] P. T. Eugster, P. Felber, R. Guerraoui and A.-M. Ker-

marrec, “The Many Faces of Publish/Subscribe,” *ACM Computing Surveys*, Vol. 35, No. 2, June 2003, pp. 114-131. doi:10.1145/857076.857078

[2] S. Oh, J.-H. Kim and G. Fox, “Real-Time Performance Analysis for Publish/Subscribe Systems,” *Future Generation Computer Systems*, Vol. 26, No. 3, March 2010, pp. 318-323. doi:10.1016/j.future.2009.09.001

[3] J. Soldatos, I. Pandis, K. Stamatis, L. Polymenakos and J. L. Crowley, “Agent Based Middleware Infrastructure for Autonomous Context-Aware Ubiquitous Computing Services,” Computer Communications, Vol. 30, No. 3, February 2007, pp. 497-498.

[4] OMG, “Data Distribution Service for Real-Time Systems. V1.2,” 2007. http://www.omg.org/spec/DDS/1.2/PDF/

[5] OMG, “The Real-Time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol Specification. V2.1,” 2009. http://www.omg.org/spec/DDSI/2.1/PDF/

[6] F. Ronci and M. Listanti, “Service Oriented Middleware Solutions for Emergency Communication Networks,” In: D. Giusto, A. Iera, G. Morabito and L. Atzori, Eds., *The Internet of Things*, Springer-Verlag Inc., New York, 2010, pp. 141-153. doi:10.1007/978-1-4419-1674-7_14

