Computer Networks 56 (2012) 1215–1235

Contents lists available at SciVerse ScienceDirect

Computer Networks

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

On data dissemination for large-scale complex critical infrastructures

Marcello Cinque [a], Catello Di Martino [a], Christian Esposito [a,b,*]

[a] Dipartimento di Informatica e Sistemistica (DIS), Università di Napoli Federico II, via Claudio 25, Napoli 80125, Italy
[b] Laboratory iTem "Carlo Savy" of the Consorzio Inter-universitario Nazionale per l'Informatica (CINI), Campus Monte S. Angelo, Napoli 80125, Italy

ARTICLE INFO

Article history:
Received 20 May 2011
Received in revised form 2 October 2011
Accepted 18 November 2011
Available online 8 December 2011

Keywords:
Publish/subscribe middleware
Peer-to-peer systems
Reliability assessment
Stochastic Activity Networks

ABSTRACT

Middleware plays a key role for the achievement of the mission of future large scale complex critical infrastructures, envisioned as federations of several heterogeneous systems over Internet. However, available approaches for data dissemination result still inadequate, since they are unable to scale and to jointly assure given QoS properties. In addition, the best-effort delivery strategy of Internet and the occurrence of node failures further exacerbate the correct and timely delivery of data, if the middleware is not equipped with means for tolerating such failures.

This paper presents a peer-to-peer approach for resilient and scalable data dissemination over large-scale complex critical infrastructures. The approach is based on the adoption of epidemic dissemination algorithms between peer groups, combined with the semi-active replication of group leaders to tolerate failures and assure the resilient delivery of data, despite the increasing scale and heterogeneity of the federated system. The effectiveness of the approach is shown by means of extensive simulation experiments, based on Stochastic Activity Networks.

1. Introduction

Large-scale complex critical infrastructures (LCCIs) [1] are emerging as a new paradigm to build future world-wide monitor and control systems (MCSs) [2]. An LCCI consists of an Internet-scale interconnection of heterogeneous, sometimes already existing, systems, glued together into a federated and open system by a data distribution middleware. Concrete examples are the novel framework for Air Traffic Management under development in Europe by EUROCONTROL called "*Single European Sky ATM Research*" (SESAR) [3], and the collaborative effort of the US Department of Energy (DOE) and the North American Electric Reliability Corporation (NERC) called "*North American Synchro-Phasor Initiative*" (NASPI) [4]. In these cases, it is not practical to deploy a dedicated and proprietary network between Air Traffic Management (ATM) entities in SESAR, or Phasor Measurements Units (PMU) in NASPI. Therefore, communication is realized by means of available network infrastructures and by using IP-based protocols. As a concrete example, interaction among ATM entities will be realized on top of Pan-European Network Service (PENS) [5], which aligns with SESAR Implementing Rules and industry standard services by providing a common IP-based network service across the European region by means of several virtual private networks (VPNs) with Gold Class of Service.

The novel requirements imposed by such world-wide interconnected systems of systems cannot be fulfilled adopting the traditional architectural model of MCSs. Currently, large MCSs are conceived as monolithic and "closed world" architectures, where the overall system is fragmented in several islands of control, each focused on an assigned portion of the infrastructure and with limited collaboration with the other ones. Since each fragment takes control decisions without considering the state of

* Corresponding author at: Dipartimento di Informatica e Sistemistica (DIS), Università di Napoli Federico II, via Claudio 25, Napoli 80125, Italy. Tel.: +39 081 7683874/676770.

E-mail addresses: macinque@unina.it (M. Cinque), catello.dimartino@unina.it (C. Di Martino), christian.esposito@unina.it (C. Esposito).

1389-1286/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2011.11.016

the others, the final control decisions are not optimal. Therefore, the novel LCCI perspective pursues the possibility to let fragments orchestrating their control decisions by an intensive information sharing conveyed by the Internet.

However, the federated architectural model envisioned for LCCIs pushes the frontiers of current technologies by posing new challenging requirements:

- High scalability: the ultra large scale of the infrastructure in terms of generated traffic load or interconnected entities must not compromise the dissemination quality;
- Interoperability: heterogeneous entities must be able to communicate among each other;
- Resiliency: messages must be delivered to all the interested destinations, even in the presence of node and/or link failures [6].

It is still unclear what underlying data distribution model should be adopted in LCCIs, since none of the current solutions are able to satisfy all the mentioned requirements. Today, a promising solution to obtain scalability is represented by middleware infrastructures adopting the publish/subscribe interaction model [7], characterized by natural decoupling properties among interacting parties. At the same time, interoperability issues are being faced by adopting standardized solutions, such as the OMG *Data Distribution Service* (DDS) for publish/subscribe services [8], as demonstrated by its track record of industrial deployments in mission- and business-critical systems. The *Real Time Publish/Subscribe* (RTPS), provides a standardized message exchanging protocol in the DDS standard. For instance, EUROCONTROL has selected DDS as reference technology for the SESAR project. Despite its advantages, DDS is not a viable solution for LCCIs since it is not able to guarantee both scalability and resiliency in the context of large-scale federated systems. In fact, RTPS presents the following issues: (i) it uses an *Automatic Repeat reQuest* (ARQ) scheme for guaranteeing data delivery resiliency by means of message buffering and retransmission, which presents scalability issues, such as implosion and exposure, and does not provide full resiliency due to buffer overflow problems [9]; and (ii) it adopts a decentralized unbrokered architecture, based on IP Multicast, which is known to exhibit severe deployment limitations over the Internet [10,11].

The contribution of this paper is twofold. First, we present TODAI (Two-tier Organization for DAta dissemination Infrastructure), a novel data dissemination scheme for DDS-compliant middleware, which addresses the mentioned issues, and contributes to the on-going discussion about the improvement of RTPS. Second, we develop a set of detailed performability models [12] for Publish/Subscribe LCCIs, used to compare TODAI and RTPS in terms of their performance and resiliency to node and link failures. The paper extends our previous research [13] providing further details on TODAI, proposing the performability models and the failure assumptions defined to conduct the analysis, and providing additional experimental results.

The proposed approach is based on a peer-to-peer (P2P) model, which is able to provide high data delivery resiliency in Internet-scale infrastructures, such as LCCIs, while inheriting the attractive scalability properties of publish/subscribe services and interoperability characteristics of the DDS. Specifically, the proposed paradigm (i) adopts a *super-peer* architecture, to handle the LCCI federated structure; (ii) it implements a *semi-active replication strategy*, to reduce the probability that a group of peers is unreachable due to node crashes; and (iii) it uses an *epidemic algorithm* to implement a proactive and reliable Internet-scale multicasting service. In fact, it has been illustrated how epidemic forwarding can be used to build scalable and reliable communication systems [14]; while, it has been empirically proved that epidemic multicast causes balanced overhead distribution among receiving peers and it is scalable as group size, publishing rate and network failure rate increase [15].

TODAI and RTPS are modeled using the Stochastic Activity Networks (SANs) [16] formalism. The proposed models allow to evaluate several figures of interests such as: (i) super-peer availability, (ii) resiliency to node crashes and link failures during the data delivery process, (iii) dissemination latency, and (iv) dissemination overhead, while varying several parameters, such as the failure rate, the gossiping fan-out, the data publishing rate, and the number of nodes. For instance, the proposed models allow us to point out that TODAI is able to deliver a resiliency level up to 99.999% over a year, while keeping a delivery latency of 87 ms, against a resiliency of 99.9% and a latency of 62 ms in the case of RTPS, in the same simulated scenario. Finally, the performed simulations allow us to investigate the limits of both solutions, and they confirm the limitations of RTPS when applied to LCCIs, in terms of buffer overflow issues due to the adopted ARQ scheme.

The paper is structured as follows: Section 2 introduces assumptions and middleware requirements for LCCIs and it includes a description of the background, with a brief presentation of the RTPS protocol; Section 3 describes the problem statement, by highlighting the limitations of RTPS when applied to LCCIs; Section 4 describes in details the proposed approach; Section 5 illustrates key concepts of SANs and presents the models we have developed to perform the simulation based performance analysis, while Section 6 describes the obtained results; Section 7 presents the related work; last Section 8 concludes the paper with final remarks and future directions for the presented work.

2. Background and requirements

2.1. LCCI

An LCCI, depicted in Fig. 1(a), is composed of several systems interconnected by means of wide-area networks, such as the Internet; therefore, it represents an example of a *Ultra Large Scale* (ULS) system [17]. LCCIs represent a solution to overcome the unsuitability of traditional architectures, and to satisfy the urgent need for a more integrated control architecture. In fact, the federation that

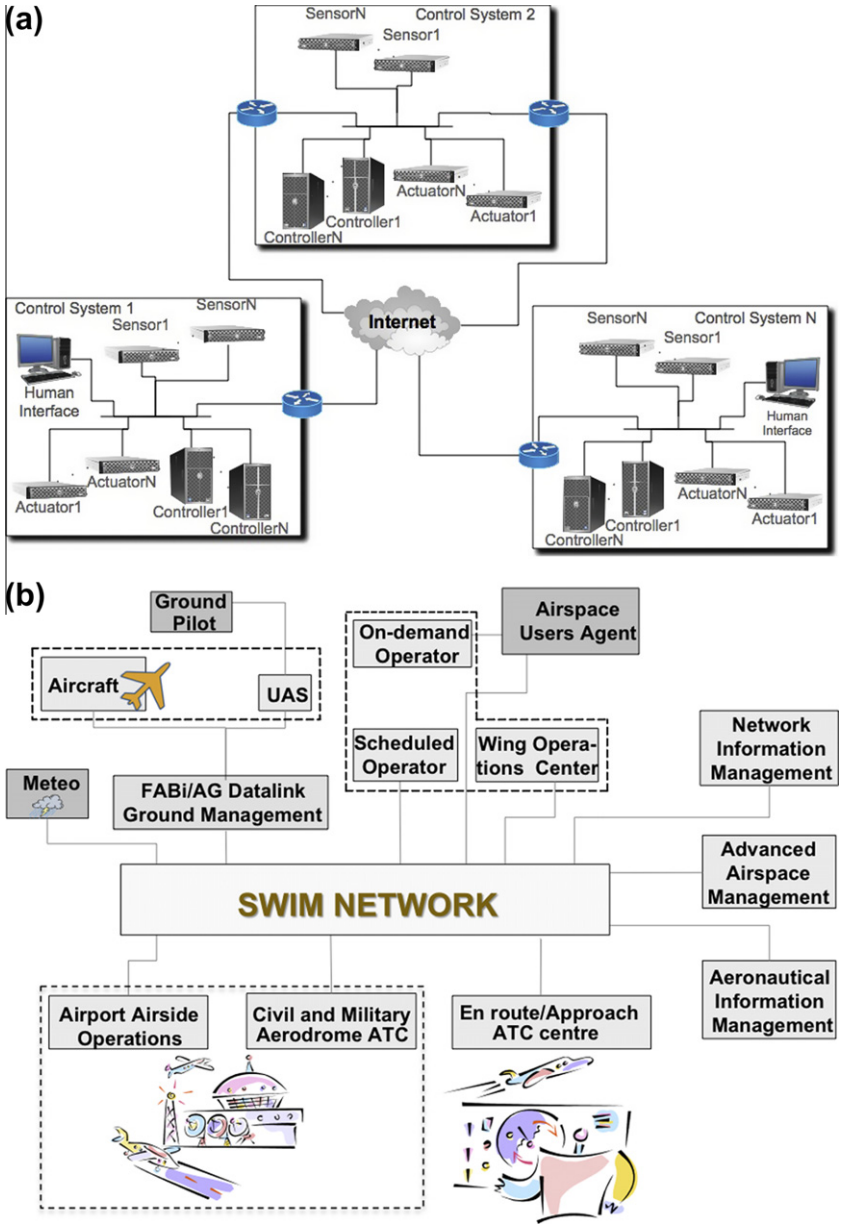

Fig. 1. LCCI as a federation of systems (a), an example of LCCI: SESAR architecture as a federation of elements interconnected by SWIM (b).

characterizes LCCIs goes beyond the trivial exchanges of monitoring data between distinct control systems, but allows implementing complex decentralized and distributed control algorithms where decisions are taken by the cooperation of several geographically distributed controllers. The collaborative intelligence underlying the control decision-making process is necessary since control decisions taken in a given portion of a critical infrastructure may affect the other portions.

In an LCCI each system is deployed on a distinct routing domain controlled by a different organization, without a central management center for the overall LCCI, and a middleware solution is used for interconnecting all the systems by conveying data over a wide-area network. Many of the ideas behind LCCIs are increasingly developing in several current projects that aim to define innovative critical systems. As mentioned, SESAR represents the best concrete example of LCCI, where all the ATM entities, such as Airports Managers, En route/Approach Air Traffic Control (ATC) centers, Regional/National/European Airspace Management or even Aircrafts, are interconnected by means of a middleware under development, named

System Wide Information Management (SWIM), as illustrated in Fig. 1(b).

2.2. Middleware requirements for LCCI

The LCCI data dissemination middleware plays a key role, since it directly affects the mission of the overall infrastructure. In general, due to the criticality of LCCI applications, the adopted middleware has to satisfy the following requirements: *Scalability*, *Interoperability* and *Resiliency*. Scalability requirements derive from the tight cooperation needed among the different systems federated in a LCCI, which involves managing and moving massive amounts of data over wide-area networks and among several destinations. In particular, the delivery latency should not be strongly affected by the number of interested destinations and by the traffic due to exchanged data. Interoperability requires that all interacting parties can communicate each other, despite heterogeneity factors (*e.g.*, adopted operating systems, programming languages, hardware architectures, etc.), and it is usually achieved by using standardized middleware solutions and by developing software wrappers to integrate legacy applications. Finally, resiliency requires the disseminated information to reach all intended destinations, despite the occurrence of node and/or link failures, and sporadic packet losses.

2.3. Middleware solutions for LCCI

Commercial-off-the-shelf (COTS) components, based on *Distributed Object Computing* (DOC) middleware, such as the *Common Object Request Broker Architecture* (CORBA), are widely used in traditional critical systems. Instead of the conventional request/reply-based technologies, *Distributed Event-Based Systems* (DEBS) [18] adopts a *Publish/Subscribe* communication model [7] that provides decoupling in time, space and synchronism among the participants of a communication, hence enforcing scalability. This communication model is known to be more efficient in terms of latency and throughput, in the case of periodic, multi-point data exchanges, and to be able to drastically reduce the network overhead [7]. Publish/subscribe maps well to connectionless protocols, and can take advantages of multicast technology to efficiently support one-to-many interactions. The event-based style carries the potential for easy integration of autonomous, heterogeneous components into complex systems that are easy to evolve and scale. In view of the above considerations, the use of publish/subscribe is preferable to request/reply in many information-driven scenarios, and result suitable for the scalable integration of mission critical systems, as foreseen by LCCIs. A significant amount of publish/subscribe systems has been developed and implemented in the last years, both by industry and by academia [19]. Most of publish/subscribe services lack the necessary support for mission critical systems. The main weaknesses of these solutions are related to either a limited or not existing support for Quality-of-Service (QoS), or to the lack of architectural properties to reach resiliency on a large scale [20].

Algorithm 1. Data dissemination among RTPS applications

```
multicast (msg):
 1: send_buf = send_buf ∪ msg;
 2: if msg.id == I then
 3:    msg = msg + hb;
 4: end if
 5: send_to (msg, M);
receive_msg (msg):
 1: receive_buf = receive_buf ∪ msg;
 2: id = deliver(receive_buf);
 3: if msg == msg + hb then
 4:    send_to(id + 1,msg.src);
 5: end if
```

Algorithm 2. RTPS Ack and buffer management

```
receive_ack (ack):
 1: id = ack.id;
 2: if ∃send_buf[id] then
 3:    for all send_buf[i] ∀i ⩽ id do
 4:       send_buf[i].acked = send_buf[i].acked + 1;
 5:    end for
 6:    msg = send_buf[id];
 7: end if
 8: send_to(msg,ack.src);
 9: refresh(send_buf);
refresh (buf):
 1: for i = 0 to buf.size then
 2:    if buf[i].acked == M.size then
 3:       buf = buf − buf[i];
 4:    end if
 5: end for
```

2.4. Data Dissemination Service (DDS)

Recently, in order to fill the QoS gap in standards for publish/subscribe services, OMG adopted a new specification, called *Data Distribution Service* (DDS) [8]. It aims to provide a standardized solution to data distribution in various types of real-time applications, balancing predictable behavior and implementation interoperability, efficiency, and performance. It adopts the Publish-Subscribe pattern and a *Data-centric View*, *i.e.*, routing is not performed based on destination addresses (Address-centric View) but on the type of data to deliver to interested destinations. It relies on the use of several QoS parameters to adapt the middleware behavior in order to meet different application requirements, and to tune the robustness of the middleware against the network unavailability and the information timeliness. The DDS standard does not define the message-exchange protocol to be used by implementations, so OMG defined a standard ''wire protocol'', called *Real-Time Publish Subscribe* (RTPS) [21]. In particular, its main features include performance and QoS properties to enable best-effort and reliable communication in standard IP networks over multicast or unicast connectionless best-effort transport protocols such as UDP/IP. This specification relies on a reliable multicast for the event dissemination

where a publisher is directly linked with all the interested subscribers by means of IP Multicast.

RTPS adopts a particular retransmission-based scheme to achieve reliability, called *Selective Repeat ARQ* [22], which has been proved to outperform all the other ARQ variations [23], and illustrated in Algorithms 1 and 2:

- *Piggybacking HeartBeat*: the publisher executes the *multicast* operation for each message by storing the message in a sending queue (row 1), including a particular command, namely HeartBeat (HB), every *I* (a parameter set by the user) messages (row 3), and sending the message within the group (row 5);
- *NACKACK*: the subscriber invokes the *receive_msg* operation for each received message. Specifically, it stores the message in a receiving queue (row 1), delivers to the application all the messages in the queue that are in order and without gaps in the sequence numbers (row 2), and sends a NACKACK message to the publisher indicating the sequence number of the earliest message it has not received (row 4). The NACKACK serves two purposes: (i) sending an acknowledgement for the notifications that have been stored into the received queue such that the publisher knows the state of the subscriber; and (ii) requesting any missed notification;
- *Retransmission*: the publisher executes the *receive_ack* operation upon the reception of a NACKACK, which triggers a retransmission of the requested message if it is still stored in the sending queue (row 8) and updates the sending queue by invoking the *refresh* operation (row 9). Such operation checks if a given message has been acknowledged by all the members of the group (row 2), and deletes the message from the queue in the positive case (row 3).

3. Problem statement

A first benchmarking work [24] demonstrated that OMG DDS-compliant implementations perform significantly better than other publish/subscribe implementations in terms of performance and reliability. However, the DDS specification is affected by flaws that limit its applicability to LCCIs.

DDS is based on IP Multicast, and this is generally referred as a point of strength, since it enables DDS implementations to deliver data with low latencies [25]. However, this choice has the weakness of limiting the usability of DDS in large-scale infrastructures due to the well-known deployability and scalability limits of IP Multicast over the Internet [10]. Currently, IP Multicast is supported only within few and scattered "islands", while the other portions do not support it. Connectivity among routers supporting IP Multicast can be provided using point-to-point IP encapsulated tunnels [26]. However, such solution exhibits severe reliability limitations, *i.e.*, it strongly results vulnerable to the failures of the routers at the end and along the tunnel. This solution also suffers of maintainability issues, *i.e.*, the tunnel needs to be manually re-established by human operators every time a failure occurs. In addition, IP Multicast exhibits a severe performance impact on routers and NIC hardware, which may fail to filter incoming messages beyond a few dozen multicast groups [27]. Finally, IP Multicast has no, or limited, regulation mechanisms for the traffic exchanged over the network [28]. This, in turn, may cause overloading of network resources and the consequent increase of message losses experienced by applications.

The ARQ-based scheme adopted by RTPS, as well as any other ARQ-based scheme, is known to have scalability and resiliency limitations when the number of destinations grows and the message loss pattern experienced by the network exacerbates [29]. To better support such claims, we have conducted a performance study of a real-world DDS implementation on an Internet-representative testbed. The interested reader can find the details of the study in [9]. Here, we briefly report the conclusions on a generic ARQ scheme we have drawn based on the achieved results:

- The latency of a given message is a function of the delivery time of the previous ones.
- When the publishing rate is high, the previous relation can lead to the instability of the sending queue of the publisher (*i.e.*, the queue grows indefinitely, and can lead to buffer overflow and message loss problems).
- The instability phenomenon of the sending queue is also related to the network conditions: worse network conditions not only affect the performance of the middleware, but also its predictability, since the measured latency exhibits large fluctuations.

Finally, the retransmission technique adopted by DDS, and, in general, by publish/subscribe services, is not adaptive with respect to the heterogeneous conditions of the network. DDS specification states, in the Platform Specific Model (PSM) for UDP/IP connections, that ARQ has to be used when the users require a reliable communication, but nothing is said about network conditions. In the typical situation of an LCCI illustrated in [1], DDS would use ARQ even when not needed, *e.g.* when the network is not exhibiting any loss pattern.

4. The TODAI approach to data dissemination

The proposed dissemination approach is based on the federated organization of LCCIs. In general, we can distinguish two distinct system domains within a LCCI: the (i) *interior-system domain*, *i.e.*, the interconnection of all the nodes within a single system, which is managed by a central organization and guarantees specific QoS policies such as reliability, and the (ii) *exterior-system domain*, *i.e.*, the wide-area interconnection of all the systems within the LCCI, which is not managed by any organization, provides only best-effort dissemination mechanisms over the Internet, and suffers of several networking anomalies (studies have proved that Internet is characterized by a resiliency between 95% and a little over 99% [30] and does not provide any guarantees on the offered Quality-of-Service (QoS) [31]). The middleware will be tailored on the features of these two different domains, *e.g.*, IP Multicast may be a winning choice for the interior-system, but not

for the exterior-system, or resiliency to networking failures is not an issue for the interior-system, while it is crucial for the exterior-system.

Starting from these considerations, we propose a novel organization of a publish/subscribe service for LCCIs named "Two-tier Organization for DAta dissemination Infrastructure" (TODAI), which is able to resolve the limits presented in the previous section and allows fragmenting the issue of achieving effective data dissemination in the overall LCCI in two main subproblems, which can be treated separately. Specifically, our driving idea consist of (i) applying a hybrid peer-to-peer organization of the LCCI, as described in SubSection 4.1, so to have a two-layered organization of the middleware in *peer groups* (for the interior-system domain) interconnected by an overlay of *super-peers*, one for each group and communicating each other on the exterior-system domain; (ii) adopting a semi-active replication of the super-peer for each peer group, as illustrated in SubSection 4.2, in order to avoid isolations of a peer group in case of super-peer crashes; and (iii) using an epidemic algorithm for the resilient communication among peer groups within the exterior-system domain, avoiding non-scalable retransmission schemes, as illustrated in SubSection 4.3.

4.1. Super-peer organization

The assumed two-layer peer-to-peer architecture consists of two types of nodes: (i) peers, which advertise and publish their own data and/or subscribe to data owned by other peers, and (ii) super-peers, which are special peers also in charge of interconnecting groups of peers. Fig. 2 shows the architecture of the overall LCCI structured in two distinct levels: (i) peer groups, which cluster all the peers that reside on the same interior-system domain, *e.g.*, a Local Area Network (LAN), and which contain at least one super-peer; and (ii) a super-peers group, which represents the interconnection of the super-peers in the exterior-system domain. This organization reflects the fragmented nature of LCCIs: the peer groups represent the systems composing an LCCI as shown in Fig. 1(a), while the super-peer group abstracts the network that federates them.

While peer groups exhibit a hybrid peer-to-peer topology managed by the super-peer, the super-peer group is organized as a pure peer-to-peer system by using an Internet-scale overlay. There is a considerable amount of literature focused on peer-to-peer application-level multicast, and the on-going debate in this field is to determine the right approach, in terms of performance and reliability, to structure the participants in a multicast session [32]. We decided to structure the super-peer group according to a *mesh-based* approach, which exposes a less structured organization by letting each node to employ a *swarming delivery mechanism* to a certain subset of nodes. Such an unstructured topology has the strength to enforce resiliency since no node plays a key role and its unavailability does not affect the correctness of the adopted communication protocol.

Algorithm 3. Bootstrap of a peer

```
boot_node (id):
 1: is_leader = bully_election(id,Leader);
 2: id_replica = −1;
 3: if is_leader == false then
 4:   for all i < R do
 5:     is_replica[i] = bully_election(id,Replica[i]);
 6:     if is_replica[i] == true then
 7:       for all j > i do
 8:         is_replica[j] == false
 9:       end for
10:       id_replica = i;
11:       break;
12:     end if
13:   end for
14: else
15:   Replica = false;
16:   run_leader(id);
17: end if
18: if id_replica! = −1 then
19:   run_peer(true);
20: else
21:   run_peer(false);
22: end if
```

Algorithm 4. Execution of a Peer

```
run_peer (replica):
 1: while true do
 2:   wait_on_event(msg,T);
 3:   if msg == null && replica == true then
 4:     is_leader = bully_election(id,Leader);
 5:     if is_leader == true then
 6:       Replica[i] = false;
 7:       run_super_peer();
 8:     end if
 9:   end if
10:   if is_from_upper_layer(msg) == true then
11:     group = msg.destination;
12:     msg.type = app;
13:     send_to(msg,group);
14:     msg.type = rep;
15:     send_to(msg,Replica ∪ Leader);
16:   else
17:     if msg.type == rep && replica == true then
18:       queue = queue ∪ msg;
19:     end if
20:     if msg.type == app then
21:       deliver(msg);
22:     end if
23:   end if
24: end while
```

4.2. Semi-active replication

We impose that peers belonging to the same group reside on the same routing domain, which provides mechanisms to support Quality-of-Service (QoS). Therefore, no additional techniques are needed to provide resiliency in the interior

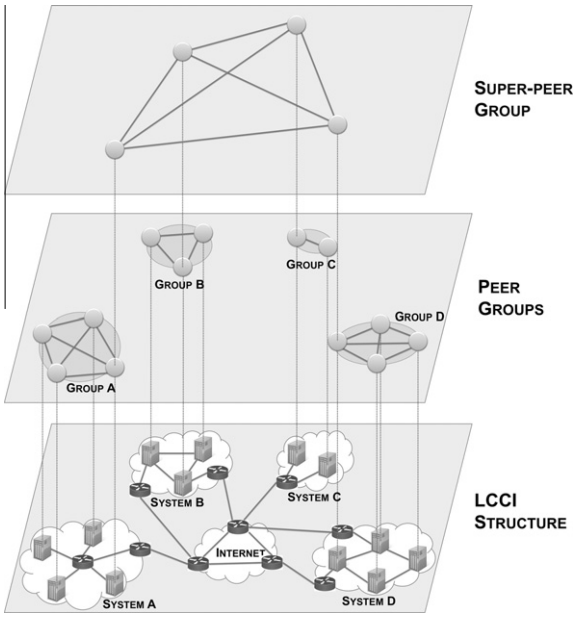

Fig. 2. Different layers of abstraction in an Internet-scale data dissemination infrastructure for LCCIs.

system domain. However, node crashes can compromise the effectiveness of communications within a group: if the super-peer is no more available, the system loses its ability to communicate with the outside world and becomes isolated. To increase the resiliency of the super-peer and to reduce the isolation probability of groups, we introduce redundancy into the design of peer groups by means of a semi-active replication strategy [33]. This replication schema consists of a leader and several followers, which act as replicas: only the leader plays the role of super-peer within the group and sends the received messages to the outside world, while the followers carry out autonomously the same computations as the leader, but do not communicate with the exterior domain. The semi-active replication has been chosen due to its low overhead to replace a failed leader and the absence of the non-determinism limitations suffered by the active replication.

In our data distribution approach, given a set of peers that belong to a group, the first activated peer is elected as leader, the next R peers are followers, where R is the replication degree, while the remaining peers can take part to the replication strategy only after the failure of the leader or one of the replicas. The election process is carried out by the well-known *Bully Algorithm* since it is able to elect exactly one coordinator in a group of distributed processes and ensures good fault-tolerance [34]. The detection of leader crashes is performed by means of keep-alive messages. Specifically, the leader periodically forwards a Keep_Alive message to all its followers; if such a message is not received within a certain timeout, the leader is suspected crashed by its followers. Then, an election process is started and the follower with the highest peer_id is elected as leader. Similarly, to keep constant the number of followers, another election process is started to elect a new follower among peers. Algorithm 3 describes the booting phase of a peer, where row 1 indicates the election of the leader. If the peer is the leader, then it will act as super-peer (row 16), otherwise it tries to be elected as replica (rows 4–13). The routine for peer will be run (rows 18–22), with a flag sets to true if it is a replica.

The execution of a peer is illustrated in Algorithm 4, where the application waits for messages until a timeout is expired. When a peer has to disseminate a given message within a certain group, it performs a multicast operation towards (i) all the peers interested to a certain class of events, and (ii) the leader and followers (described in rows 10–15). In case a message is received from an other peer, if the message has been sent to preserve consistency of the state among leader and replicas (row 17–18) then it is stored, otherwise, if the peer is interested, the message is delivered to the subscriber (rows 20–21). If no messages are received within the timeout (including keep alive messages), and the peer is a replica, then the leader is not active, and a new election is triggered (rows 3–9).

4.3. Epidemic data dissemination

The replication of super-peers alone is not enough to assure the correct communication among peer groups, which can still be compromised by message losses. A new kind of distributed algorithms has recently become popular as a solution to address scalable and resilient multicast dissemination: *Epidemic Algorithms* [35]. Their basic idea is that each process communicates periodically its knowledge about the system "states" (*i.e.*, the content of the event table of each super-peer) to a robust subset of other processes. The probabilistic and decentralized nature of these algorithms gives them some desirable properties, as demonstrated in [36]: (i) resiliency to changes in the system configuration; (ii) fault tolerance at both global (withstanding an high number of faulty nodes and message losses) and local level (not relying on the correct operation of any other specific node); (iii) simple to implement; and (iv) rather computationally inexpensive.

When a super-peer receives a new event to be disseminated within the super-peer group (as shown in Algorithm 5 at row 2), it randomly selects some known peers (the number of selected peers at each round is called *Fan-out* of the epidemic algorithm and it is decided one for all before the system is deployed), (row 7) and forwards to each of them the received event (row 8). When other super-peers receive a new event (row 2), they start a new round by sending it to some randomly-selected known peers (excluding the sender of the received event). To achieve termination of the algorithm, a new gossiping round is not issued if a super-peer receives an event that it already has stored in its event table (row 4). The gossiping algorithm does not store all messages forever in an event table, since an infinite buffer is not available. In fact, a message is removed from the event table when its liveliness is expired, *i.e.*, the total number of gossiping round passed since the event has been stored.

Algorithm 5. Data dissemination among different peer groups carried out by a super-peer

```
run_super_peer():
 1: while true do
 2:   wait_on_event(msg);
 3:   if msg! = null then
 4:     if msg ∉ buf then
 5:       buf = buf ∪ msg;
 6:       for i = 0 to FANOUT do
 7:         next_peer = ⟨Super_Peers[j]⟩_{j=Random(i)};
 8:         send_to(msg,next_peer);
 9:       end for
10:     end if
11:   end if
12: end while
```

4.4. Dissemination example

Fig. 3 provides a concrete example of the dissemination strategy applied in the proposed approach. Let us assume that node N_1 wants to publish a notification. Then, it will perform the first step by multicasting the notification, by means of an IP Multicast communication, within the cluster to all interested subscribers, plus the super-peer (indicated in figure as C_1) and its replicas (arrow 1 in figure). Upon the arrival of the notification to C_1, a second step is performed by running Algorithm 5: among the super-peers of the clusters that hold subscribers interested to the received notification, C_1 will randomly pick up some to communicate with (in the case of C_1, its fanout is 2, so two destinations are selected, *e.g.*, C_2 and C_8). When the notification will reach the chosen destinations, by means of UDP, it will be disseminated within the cluster (arrows 3′ in figure), and a new gossip round will be performed (arrows 3″ in figure). Steps 2 and 3 are repeated until the termination of the gossip algorithm, *i.e.*, all super-peers have been contacted. Let us notice that it is possible for a super-peer to be reached more than one time by a certain notification (as happens in figure for C_1 and C_8); in this case, the second notification is suppressed and no gossip round is commenced, as indicated in row 4 of Algorithm 5.

5. Modeling RTPS and TODAI behaviors

This section describes the performability models developed to compare RTPS and TODAI. Such models have been defined as a set of interconnected Stochastic Activity Networks (SAN) [16], and are solved by simulation using the MOBIUS tool [37] to estimate the reward metrics. We adopted SANs due to their flexibility and

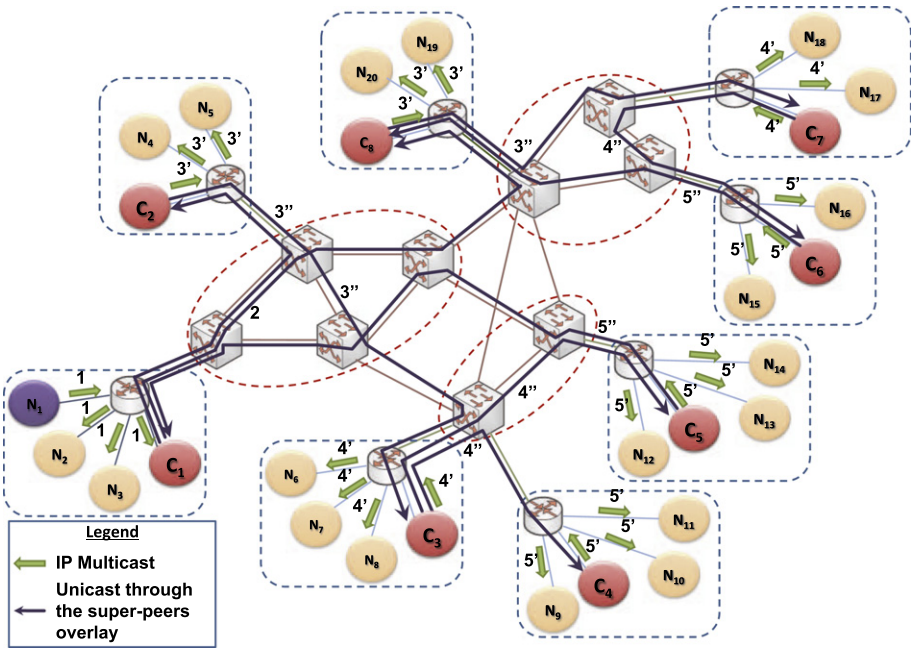

Fig. 3. Dissemination of a notification by applying TODAI.

their successful use, since the mid-1980s, for modeling the behavior of a broad range of complex systems such as Critical Infrastructures [38], railway interlocking systems [39], databases [40], supercomputers [41] and wireless sensor networks for structural monitoring [42]. SANs have a custom graphical representation consisting of places (blue/yellow circles), timed activities (thick bars) and instantaneous activities (thin bars), input gates (red triangles) and output gates (black triangles). An activity is enabled if there is a token in all the places connected to it, or if a predicate expressed into an input gate connected to it is verified. Once enabled, the amount of time to complete a timed activity (firing of an activity) can follow a specific stochastic distribution such as Exponential and Weibull. Cases can be associated to activities (represented by circles on the right side of an activity) and allow to model uncertainty upon completion of an activity. The presence of *JOIN* (which combines several distinct models) and *REP* operator (which creates several copies of a certain model) offers intrinsic support to alleviate the costs of developing models of large scale and complex systems.

SANs allow defining custom metrics by means of reward variables. The evaluation of the reward variables involves specifying a performance variable and a reward structure, which associates reward rates with state occupancies and reward impulses with state transitions, namely, a "reward" is accumulated every time events of interests happen during the simulation. In particular, we are interested in the following reward metrics to compare RTPS and TODAI:

- *Resiliency*, defined as the probability that a notification is delivered to all subscribers.
- *Overhead*, defined as the additional traffic to deliver a notification to subscribers, due to the adopted fault-tolerance mechanism.
- *Latency*, defined as the time needed to propagate a message to all the subscribers.
- *Average number of rounds*, defined as the average of rounds needed in TODAI to propagate a notification to all subscribers.
- *Average number of messages*, defined as the average number of messages exchanged during a round in TODAI.
- *Availability*, defined as the percentage of uptime of the system during a year.

5.1. Modeling assumptions

The objective of the following models is to evaluate the defined reward metrics against a set of realistic impairments. Therefore, in this work, we assume the following classes of failures:

- *Node crashes*: a node of the LCCI suddenly stops its execution, and it is restored after a certain period of time.
- *Message losses*: links may exhibit message losses, due to packet corruption, buffer overflow, and temporary drops.

We assume that all the nodes are equipped with same hardware and software, and that a unique ID is assigned to all the *SuperPeers* in the system. Publisher nodes are assumed to publish data periodically.

Failures can be related to hardware, software or the communication infrastructure (networking failures). The failure of a node occurs as service omission due to the crash of the node or the loss of connection towards other nodes [40]. Hardware faults stem from instabilities of the underlying hardware platform, and can manifest as errors at the software level [43] either in operating system or in the application. We focus on intermittent and transient faults, since it has been proved that they are by far predominant [44].

Transient hardware faults are assumed to have an exponential rate, characterized by the alternation of periods where normal fault rate is observed and periods with abnormal, higher rate. The duration of a period follows an exponential law (with normal periods being quite a bit larger than abnormal ones). The restart of the application removes the effects of the hardware faults at the application level.

Intermittent application or operating system software errors have been assumed to have an increasing rate according to a lognormal distribution [45] since this is consistent with the fact that the extent of the damage increases with time, if no recovery action is taken.

Two alternative recovery actions are encompassed: (i) restart of the application, to cure inconsistent application-level states, and (ii) reboot of the peer, to fix erroneous states of the operating system and of the applications. Both actions cause a reset of the state of the peer, including all the incoming and outgoing buffers, and a limited period in which the peer is not available.

Channel failures and message losses are modeled following the Gilbert Model which is a 1st order Markov chain model characterized by two states: state 0, with no losses, and state 1, with losses. There are four transition probabilities: (i) the probability to pass from state 0 to state 1 is called P; (ii) the probability to remain in state 0 is $(1-P)$; (iii) the probability to pass from state 1 to state 0 is called Q; and (iv) the probability to remain in state 1 is $(1-Q)$. Given the *Packet Loss Rate* (PLR), *i.e.*, the percentage of lost packets over the total number of exchanged packets, and the *Average Burst Length* (ABL), *i.e.*, the mean number of consecutive lost packets, it is possible to compute P and Q as follows [46]:

$$P = \frac{PLR \cdot Q}{1 - PLR} \quad Q = ABL^{-1} \tag{1}$$

To let the models be representative of a realistic wide-scale scenario, we have estimated PLR and ABL values by means of a network monitoring campaign performed over PlanetLab [47], a geographically distributed overlay platform designed for deploying and evaluating services over wide-area networks. Further details on the experimental campaign performed to characterize the channel are presented in [9], here we provide only brief information on such campaign. Specifically, we have selected three paths between European cities hosting PlanetLab

nodes, and measured dissemination delay, PLR and ABL (each characterized as median and inter-quartile range (IQR) of 1000 observations of 24-h monitor traces). Last, tests have been conducted by exchanging messages between the publisher and the subscribers in a round trip manner (only the last contacted subscriber return a copy of the received message to the publisher), and applying a workload extracted from the requirements of SESAR, *e.g.*, publishing rate of 100 Hz and notification size of 100 KB. Results of such campaign can be seen in Table 1, and they have been used to set PLR and ABL values as the mean of the estimates obtained over the three paths.

Table 1
Measures of network behaviour over three European paths.

	Delay (msec)		PLR		ABL	
	Median	IQR	Median	IQR	Median	IQR
Path 1	8.9	0.87	0.65	1.24	1.59	0.28
Path 2	27.16	18	1.07	4.86	1.26	0.32
Path 3	43.81	0.78	1.77	0.85	1.44	0.11

5.2. Composed models

The overall composed models are illustrated in Fig. 4, and represent the hierarchical models of RTPS, in Fig. 4(a), and of TODAI, in Fig. 4(b). They consists of several logically distinct SANs connected together through common places by the *REP* or *JOIN* operators. Failure, Recovery and Channel models are the same for both RTPS and TODAI (the names in the figure are different, *e.g.*, failureModel_pub and failureModel_sub, since Mobius does not allow to use the same name for different submodels; however the internals of the models are the same). In particular, the FailureModel and RecoveryModel mimic the failing and recovery behavior, respectively, according to the assumptions described in Section 5.1. The Channel model mimics the behavior of the communication channel, including message loss. As for PLR and ABL, the values of delay and jitter of the modeled channel are derived from the experimental campaign performed on PlanetLab.

The RTPS composed model is conceived as the join of three sub-models: (i) the channel, composed in turn by two models, one for forward communications (the same used in TODAI) and a separate one for handling the *ACKNACK* mechanism (used for RTPS only), (ii) a given number of replicated data readers, and (iii) a given number of, replicated, data writers. Reader and writer models are the join of failure and recovery modes and of the model actually performing the subscriber role (RTPS_subscriber) or publisher role (RTPS_publisher), respectively.

The TODAI composed model has a similar structure of the RTPS composed model, but it adds another subtree to model groups and super-peers. In particular, the model is conceived as the join of the Channel model (for the exterior-system domain) and the Groups model, devised as the replication of several groups. Each group is in turn composed of a group channel (for the interior-system domain), data readers and data writers, which represent simple peers (with a structure similar to RTPS nodes), and a replication of leaders for modeling the semi-active replication scheme and the behavior of super-peers. The leader subtree encompasses failure and recovery models, the Election model, mimicking the Bully election algorithm (in case of leader crash), and two models for the super-peer: one to model the initiation and the end of gossiping rounds towards other groups (TODAI_publisher) and another one to model the packet reception and forwarding process (TODAI_forwarder) when the super-peer receives a packet from another group.

In the remainder of this section, six sub-models are described in details, namely Failure and Channel models, which are shared by RTPS and TODAI models, and RTPS_publisher, RTPS_ subscriber, TODAI_publisher, and TODAI_forwarder.

5.3. Failure model

Fig. 5 depicts the SAN Failure Model used for all the modeled nodes. The model reproduces software and

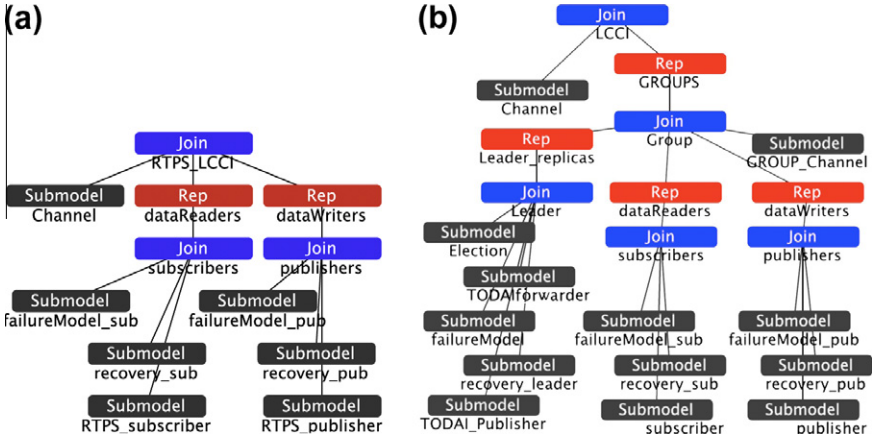

Fig. 4. Composed model of (a) RTPS, (b) TODAI.

hardware failures, according to the previous modeling assumptions. Application and OS software failures are modeled by the Lognormal distributed activities *ApplFail* and *OSFail*, respectively, whereas hardware failures are modeled by the Exponential activity *HWFail*. The rate of this last activity depends on the marking of the places *NormalHWF* (normal periods) and *burstHWF* (abnormal periods). The Exponential activities *Normal* and *Burst* govern the alternation of normal and abnormal periods. During normal periods, which have an expected duration indicated by the parameter T_N, transient hardware faults occur with a rate λ_N. On the other hand, during abnormal periods, having an expected duration T_B, faults are characterized by a higher rate λ_B. When the failure model triggers failures, the other sub-models of the node (*e.g.*, publisher and subscriber sub-models) are disabled, by using proper shared places.

5.4. Channel model

The Channel model is shown in Fig. 6(a). It models the behavior of the packet delivery process over a communication channel, including the possibility of message loss. A token placed in SampleSent (shared with publisher sub-models and with the TODAI_forwarder) enables the input gate named toTheChannel, which takes care of taking the packet from the DataWriter and moving it to the place called OnTheChannel. The latter is a circular queue, which abstracts the channel. Once packets have been placed in this queue, the value of the place PacketsOnTheChannel is incremented, and this value (if positive) triggers the timed activity called UDP_Channel. This activity has a normal distribution, whose mean and variance are determined by the size of the sent packet, stored in the place PktDim. Once terminated this activity, the packet is taken from the queue and inserted in the place named SampleIn (shared with subscriber sub-models and with the TODAI_forwarder), which indicates the reception of a valid packet from the channel. Packets can be also discarded by the UDP_Channel activity, according to the loss probability parameter P_{loss} associated with the cases of the activity.

It has to be noted that the return channel from DataReaders to the DataWriter has been modeled as a separate sub-model, shown in Fig. 6(b). This choice has been motivated by the necessity of handling the queue of received *ACKNACK* (ACKNACK_ rec) packets, to trigger retransmissions at the RTPS publisher model.

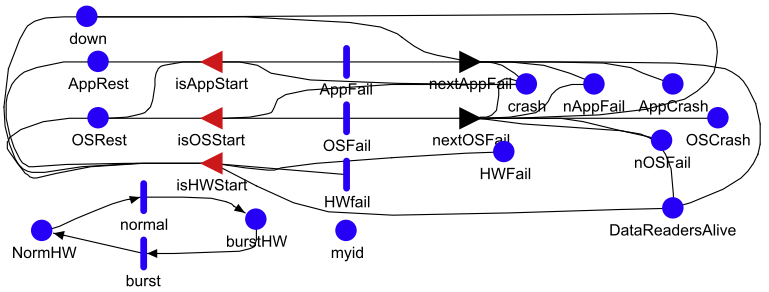

Fig. 5. Node failure model.

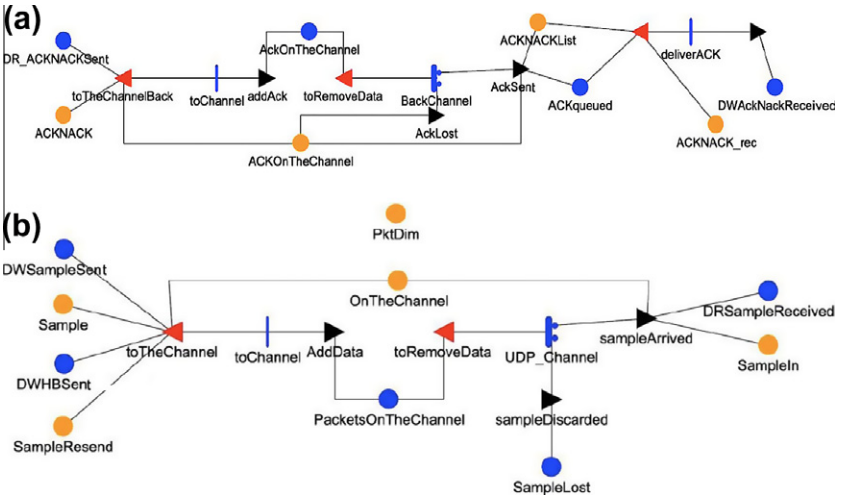

Fig. 6. Channel model: (a) forward channel (b) backward channel (used by the RTPS model only).

5.5. RTPS_publisher

The RTPS publisher model (shown in Fig. 7) mimics the publication of data (as explained in Section 2.4) from an application to a set of subscribers through a DataWriter, *i.e.*, an RTPS entity that the publisher application can use to disseminate notifications.

A mark in the place *PubAppLive* indicates that there is a publishing in progress, and it enables the input gate named *CanWrite* which is in charge of performing the actual writing in the output queue. In particular, each new published sample, to which a sequence number is added, is written in the place *Queue* containing a circular queue. Concurrently, tokens are placed in positions named *Queued* and *SendActive*. The first place indicates that a new sample has been queued; the second enables the sending activity to_send. The activity *to_send* has two possible alternative operations, selected depending on the presence of a token in place *HBsend*. For each sent message, the number of samples in *HowManySamples* is increased. If a threshold `HBperiod` is reached, a token is placed in *HBsend*, and a HB is added to the payload of the next message to send. The actual sending process is then performed by using the channel model.

NormalRate and *Burst* are used to discriminate between periods of normal publication rate against abnormal periods (bursty publication rate), using the same logic of the failure model.

The model also controls if the sending queue is full, by activating the place named *QueueFull*. When a token is put in the place *QueueFull*, the activity *toBlock* inserts a token in *Blocked*, which blocks the write queue. If the Publisher remains blocked for a period longer than `max_blocking_time`, a timeout fires and the packets contained in the queue are discarded (modeling a buffer overflow problem), else the token is removed from *Blocked* and the publication process is resumed.

An additional role of the publisher is the retransmission of packets that have not received by a DataReader. For this purpose the Publisher model keeps track of DataReaders that are still active. In particular, a maximum value of not responded HBs is defined, in order to consider a DataReader as inactive. If one of the values of waiting HBs exceeds the threshold, the publisher reduces the number of active DataReaders, which has the effect to make the publisher waiting for fewer ACKNACK to update the queue. The mechanism for emptying the sending queue is based on the number of received *ACKNACK*. If all active DataReaders sent their *ACKNACK*, then the queue can be updated by removing the acknowledged message. If a DataReader asks for retransmitting lost packets, they are immediately placed on the channel.

5.6. RTPS_subscriber

The RTPS subscriber model, illustrated in Fig. 8, mimics a simple reliable DataReader, which reads data from an input queue. The data reception is triggered by a token in the place in *DRSampleReceived* (shared with the channel model) and indicates the presence of a valid data in *SampleIn*. The model encompasses the presence of a buffer stage (place *working*) between the actual reception of the data and its queuing by inserting specific delays. Once the timer programmed by the queue operation is expired, the application accounts for samples arrived out of order and for which it must leave space in the queue in order to send *ACKNACK* when requested by a HB command. If the message contains a HB, it sends an *ACKNACK* message to the DataWriter. When a new message is received, a counter indicating the expected sequence number is incremented. Messages stored in the receiving queue can be moved to the output queue (which models the queue to applications) if they are sorted by timestamps. This operation is managed by the output gate *enqueue*, while the control of the ordering of received samples is made by the input gate *Checkorder*. The model also takes into account the depletion of node receive buffers in the case of the reboot of the DataReader (gate *flush*).

5.7. TODAI_publisher

Fig. 9 shows the TODAI super-peer model when acting as publisher. The model is responsible of triggering gossiping rounds towards the other groups. This can happen in two ways: directly, *i.e.*, the super-peer periodically creates

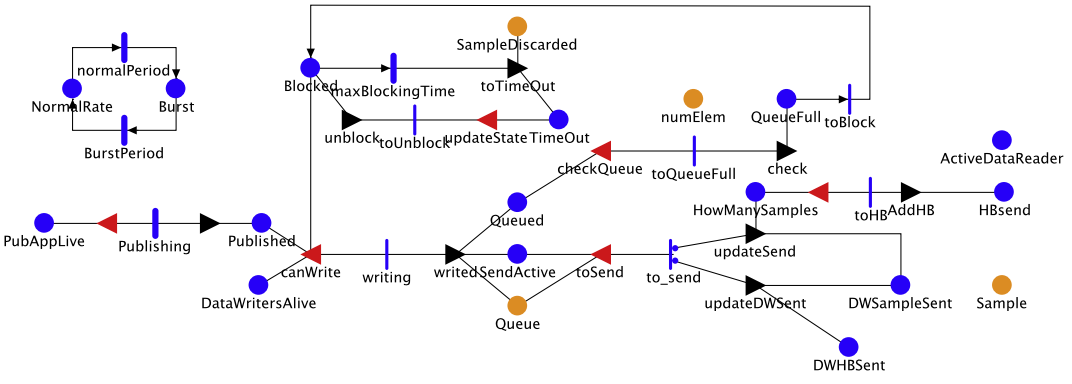

Fig. 7. RTPS publisher model.

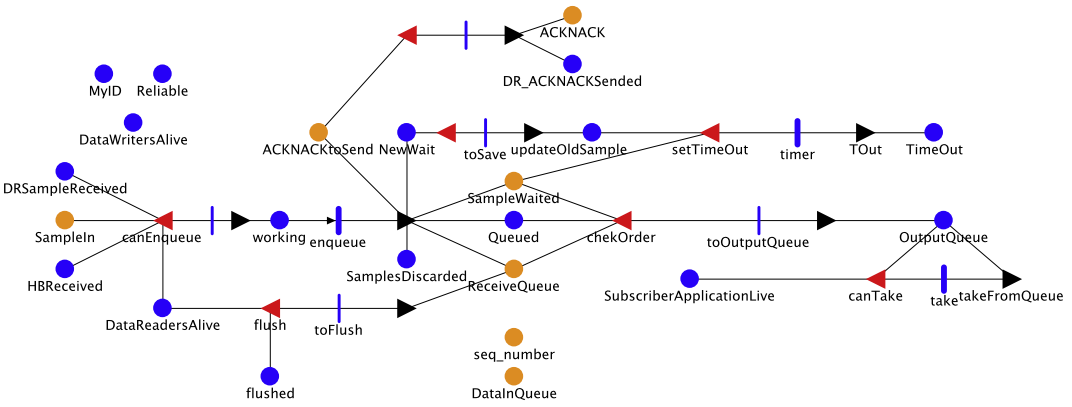

Fig. 8. RTPS subscriber model.

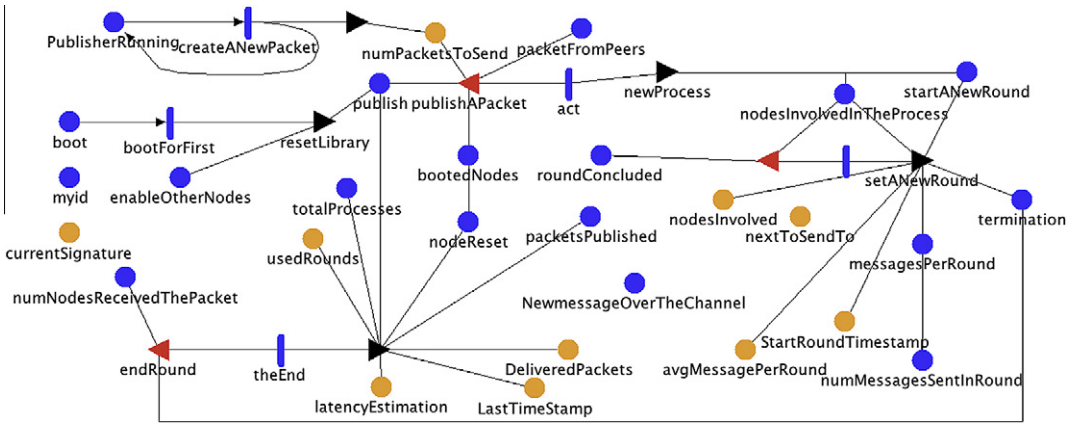

Fig. 9. TODAI publisher model.

a number of packets to be sent (it acts as a normal publisher) when a mark is present in the place *PublisherRunning*, or indirectly, *i.e.*, when a packet is received from another publisher (peer_publisher model) within the group, by means of the place *PacketsFromPeers* (shared with the channels to the peers of the group). In both cases, when at least one mark is contained in the places *numPacketToSend* and *publish*, the input gate *publishAPacket* enables the publishing of a specific number of packets (equal to the number of marks contained in *numPacketToSend*) and starts a new Gossiping Round (a mark is put in the place *startANewRound*). Packets are structured as records which contain a unique signature, copied in the place *currentSignature*. The signature is a random number selected by the publisher to identify all created packets.

Then, the IDs of the nodes to be contacted are extracted by setting the corresponding elements of a vector contained in the place *nodesInvolved*. The extraction is performed by using the gossiping, *i.e.*, for each node in the system, with a given probability (the gossiping fan-out) it decides whether or not to contact that node, by means of the output gate *setANewRound*. The selected node IDs are then stored in the extended place *nextToSendTo*, which contains a list of all the nodes to be contacted by means of the channel model. After the publishing, the number of marks of the place *publish* is set to zero.

The current round is terminated if all the contacted nodes received the published data. This is accomplished by the input gate *endRound* which activates the action *theEnd* consequently. In order to keep track of all the nodes which received the current packet, each packet is equipped with a sequence number and with a signature, stored in *currentSignature*. When a node receives the sample for the first time, the number of marks in the place *numNodesReceivedThePacket* (shared with the TODAI_forwarder model) is incremented. When the number of marks of such place reaches the number of groups, then the round is ended and new round can start.

The place *bootedNodes* is used to keep track of the active nodes. It is needed in the enabling predicate of input gate *publishAPacket*, since a packet is created only after all groups are up and running.

5.8. TODAI_forwarder

Fig. 10 shows the SAN model of the TODAI super-peer node when acting as forwarder. This model basically performs two operations: it receives messages from TODAI

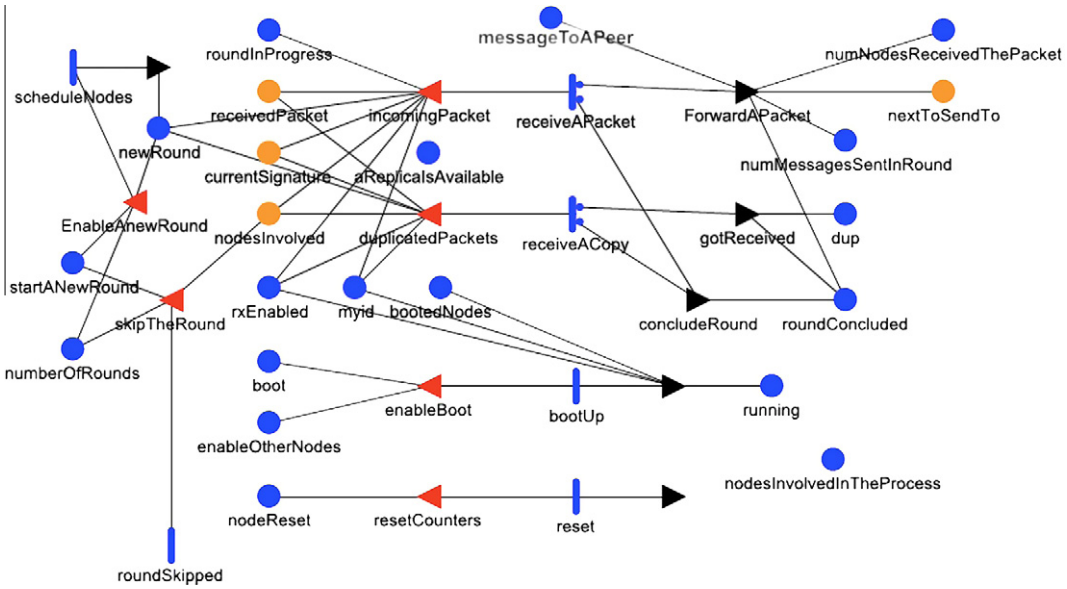

Fig. 10. TODAI forwarder model.

super-peers (*i.e.*, from the TODAI_publisher model or from other TODAI_forwarder models) and forwards them to other super-peers using the gossiping logic. To this aim it (i) is connected to the channel model and (ii) it must keep track of the packets already forwarded, to avoid the explosion of messages in the system. For (i), the model is activated once the place *receivedPacket* (shared with the Channel model) contains a new packet. Then it checks whether the packet is a duplicated packet, by using the signature of the packet. When the forwarder receives a packet for the first time, it inserts its signature in a associative table than is later used in the input gate *duplicatedPacket*. This gate extracts the signature from the place *receivedPacket* and use it as the key for the associative table. In the case of a hit, it discards the packet (action *receiveACopy*) and forces an increment of the number of marks in the place *dup* which is later used to estimate a reward metric for duplicated packets. In the case the message was not duplicated, the action *receiveAPacket* is enabled and with a probability of `l-gossipingFanOut` the packet is not forwarded and the round is concluded. Otherwise, the packet is forwarded and the gate *ForwardAPacket* is used to select randomly the nodes where to forward the message to. Selected nodes IDs are then stored in the place *nextToSend* which is shared with the channel model. When the forwarding of a packet is ready, the packet is also sent to subscribers within the group (modeled with the peer_subscriber model). To this aim, the *messageToAPeer* place is used, shared with the channels of subscribers within the group.

6. Experimental results

In this section, we report the results obtained from the simulation of the defined SAN models. The aim of the simulation campaign is to study the trade off between resiliency improvement and performance penalty exhibited by our approach and RTPS. Simulations are performed over a simulated time of 5 years, and confidence level for the measurements set to 99%.[1]

The proposed models have been used to simulate several realistic LCCIs, inspired from a real deployment named as Co-Flight [48] (a novel Flight Data Processor, realized within the context of SESAR by Thales and Selex Sistemi Integrati) with the main objective of enabling the cooperation and interconnection of all European airports and hubs. Co-Flight is an ATM application designed to update flight data plans with information received from the Radar and other ATM instrumentation, and to distribute them to control towers and other interested ATM entities. This way, we design a set of experiments with a varying number of peer groups and failure rate, in order to evaluate the sensitivity of dissemination protocols to node and/or network failures for LCCIs of different size. In particular, we decide to focus on LCCI configurations spanning from 32 to 128 groups, where each group is composed of 64 peers, and enabling the simulation of several LCCIs, ranging from 2048 to 8192 nodes, representative of the actual size of the current European airport network, shown in Fig. 11.

Simulation parameters, reported in Table 2, are mostly taken from direct measurements on a real testbed, *i.e.*, PlanetLab (http://www.planet-lab.org), and from [49]. In addition, the model has been executed considering the workload of CoFlight. The characteristics of this product are: (*i*) publishing rate of 0.1 s, (ii) about ten data consumers per given piece of information, and (iii) a message size of around 100 Kbytes.

[1] A confidence interval gives an estimated range of values which is likely to include an unknown population parameter. In our case, 99% of the samples of the estimated metrics are within the interval of ±0.5% of their expected value.

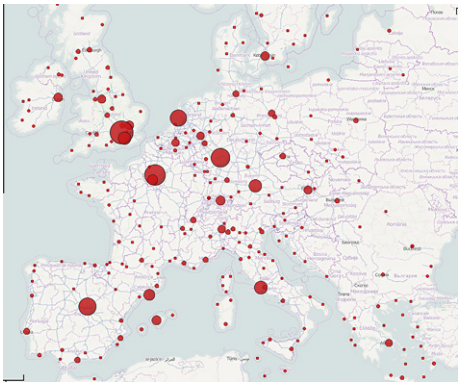

Fig. 11. Map of the European Airports and hubs (source: Wikipedia, the free encyclopedia).

6.1. RTPS

Fig. 12(a) indicates the resiliency of RTPS against the number of considered groups, with a low failure rate ($1.1E{-}07$). The resiliency is evaluated as the fraction of packets correctly delivered to all interested destinations when network suffers of message losses. We adopted a group-based infrastructure for RTPS, to make results comparable with our approach and to simulate a realistic deployment scenario over a LCCI. However, RTPS does not exploit the group-based infrastructure, since no super-peers nor replicas are defined. We assume nodes on the same group (in the interior-system domain) to communicate over reliable, lossless channels, while nodes of different groups (interconnected via the Internet on the exterior-system domain) are assumed to communicate over unreliable channels, with a packet loss probability (P_{loss}) equals to 0.01.

The maximum resiliency level (99.96%) is achieved for 32 groups (2048 nodes), and decreases down to 99.87% in the worst case, corresponding to 128 groups (8192 nodes) and with a HB period of 8 s. Hence, increasing the number of nodes affects the probability of correct message delivery. This is partially due to the ARQ protocol, which requires one-to-one communications for handling lost packets, and to the absence of replication schemes for facing node failures. In particular, the increasing number of nodes impacts on the sending queue managed by publishers, in terms of overflow probability. We experienced a peak overflow probability of 80% when sending messages to 128 groups, and with a sending queue size of 1200 messages. This strongly impacts on the resiliency of the LCCI, since nodes not receiving messages cannot benefit of the ARQ scheme due to buffer overflow at the publisher side. At the same time, increasing the buffer size of all the publishers impacts on the scalability of the system. For instance, in our case, to keep the same resiliency level between 64 and 128 groups, we had to use a sending queue size of 1,440,000 packets, which corresponds to about 141 MB of required memory, for each publisher. This highlights the scalability limits of RTPS when applied to LCCIs.

For the performed experiments, the latency required for disseminating all the messages to all subscribers (not shown for lack of space) accounts to 40 ms on average, with a peak of 62.4 ms for the case of low failure rate and 128 groups.

6.2. TODAI

Fig. 12(b) indicates the resiliency of the proposed approach, against the number of considered groups. We observe a different dynamic if compared to RTPS. In particular, for the proposed approach the resiliency is an increasing function of the number of groups, while it is a decreasing function for RTPS. More specifically, the resiliency varies from 99.7176% in the worst case (32 groups, high failure rate, 3% fan-out) up to 99.9990% in the case of 128 groups, low failure rate, and 12% fan-out.

This is the effect of the proactive nature of the epidemic data dissemination, which causes the redundancy of data in the LCCI to increase proportionally to the number of involved groups, without affecting super-peers (in this case, they do not have to manage retransmissions). This effect is also observed when increasing the failure rate of 2 orders of magnitude, from $1.1E{-}07$, to $1.1E{-}05$.

As expected, we can observe how the fan-out can be used as a mean to increase the resiliency of the LCCI. For instance, looking at Fig. 12(b), for the case of 64 groups, with high failure rate, the resiliency shifts from 99.8697% up to 99.9846% when doubling the fan-out from 3% to 6%, respectively. This effect is less observable in the case of low failure rate, in which, for the same case, a 3% fan-out is already enough to deliver 99.9753% of resiliency.

The increasing level of resiliency is achieved at the price of an increased overhead, as can be observed looking at

Table 2
Simulation parameters. Sensitivity analysis are performed with respect to values reported in square brackets.

Parameter	Description	Values
λ_N	Hardware failure rate during normal periods	[1.1E−07; 1.1E−05]
T_N	Duration of the normal periods	27215640 s
λ_B	Hardware failure rate during abnormal periods	8.33E-03
T_B	Duration of the abnormal periods	360 s
P_{loss}	Packet loss probability	0.01
N_g	Number of groups	[32; 64; 128]
F	Fan-out of the epidemic algorithm	[3%; 6%; 12%]
HB	RTPS Heartbeat period	[2; 4; 8] sec
P_{rate}	Rate of the publication process	100 Hz
P_{size}	Packet size	100 KB

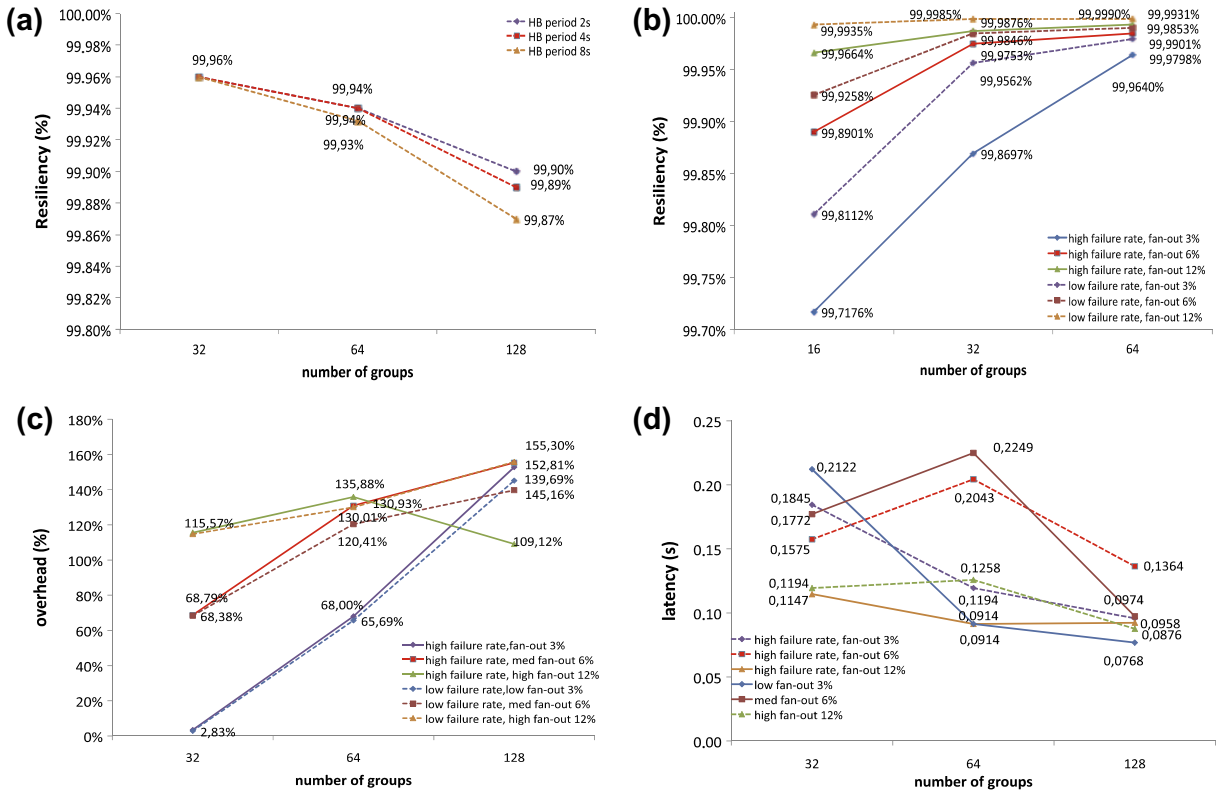

Fig. 12. Experimental results, (a) RTPS resiliency; (b) TODAI resiliency; (c) TODAI overhead; (d) TODAI latency. Dashed lines report results obtained with a low failure rate.

Fig. 12(c). The overhead, or link stress, is evaluated in terms of extra-packets needed to disseminate a single message to all the subscribers. Results show an overhead highly varying from 2.83% up to 155.30%. In almost all cases, the overhead is an increasing function of the number of groups and of the fan-out, as expected. However, we can observe that, in the case of high fan-out and high failure rate, the overhead decreases from 135.88% for 64 groups to 109.12% for 128 groups. This is the effect of failures, which increase the downtime of super-peers, and reduce the rounds needed to reach all alive super-peers.

The achieved overhead is higher than the one of RTPS, where a 25% overhead is observed to achieve a 99.9753% resiliency level in the case of 64 groups and low failure rate. The same case with the epidemic dissemination accounts for a 65.69% overhead. This is reflected by increased delivery latency. As shown in Fig. 12(d), the latency is always greater than the one achieved for RTPS (at most of one order of magnitude). However, it is interesting to note that the latency is a decreasing function of the number of nodes, thanks to the increasing number of nodes contacted for each round, whereas it is an increasing function of the number of nodes for RTPS, once again confirming the scalability properties of our approach.

Fig. 13(a) shows that increasing the fan-out, the number of needed rounds to deliver a notification decreases, as evident for the cases with low failure rate (dashed lines in the figure). With respect to the cases with high failure rate, we can observe this trend when passing from a fanout of 3% to one of 6%, while with 12% we have a number of rounds similar to the one with 6% when the group size is 32, and to the one with 3% in the other cases.

The number of messages exchanged during a round (depicted in Fig. 13(b)) strongly depends on the number of groups and the fanout. In addition, also the failure rate exhibited by the network affects the number of exchanged messages, as can be observed if comparing the lines with fanout equals to 12%.

6.3. Group availability

Finally, we performed two simulations to evaluate the trend of the system availability as the function of (i) the publishing rate and the number of nodes in a group (where each node acts as a passive replica of the super-peer, without semi-active replication), and (ii) the number of SuperPeers in the local group (*i.e.*, when the semi-active replication scheme is adopted).

The results of the first simulation are reported in Table 3. They show how the system availability increases as the number of peers grows: with the constant publishing rate of 0.1 s, using one peer we have an availability of three nines (*i.e.*, colloquial term used in engineering to indicate reliability and preceded by a number indicating the degree of such reliability, *e.g.*, electricity that is delivered without interruptions – blackouts, brownouts or surges – 99.999%

of the time would have 5 nines reliability); with five peers to 15 peers, we obtain 5 nines. It is worth noting that the availability improvement is remarkable when we pass from one to five peers in a local group. After five peers, the addiction of further peers does not lead to significant improvements. This means that a high availability level can be achieved with a relatively low amount of passive replicas (*i.e.*, backup nodes) in the group, and thus with an acceptable performance penalty. From Table 3, it is evident that decreasing the publishing rate, the availability level increases. Clearly, when the publishing rate is lower, unavailability periods becomes less perceivable by the other peers in the system or, in other terms, a lower fraction of messages are lost during unavailability periods if compared with the case of high publishing rate.

The results of the second simulation are shown in Table 4. Simulation parameters are five peers into the local group and a publishing rate of 1 s. Doubling from one SuperPeer to two, the availability improves from five nines to seven, and it remains the same with three SuperPeers. Comparing this result with the previous one, it is clear that adding one more super peer to a local group, using the semi-active replication scheme (which implies a lower time-to-repair, since the replica can suddenly replace the failed super-peer), results on a better availability level than the one achievable in the case with one SuperPeer and 10–15 peer nodes.

Table 3
Analysis of the system availability (expressed as number of nines) as function of the number of nodes and the publishing rate.

Number of nodes	System availability		
	Rate = 0.1 s	Rate = 1 s	Rate = 10 s
1	3 nines	3 nines	3 nines
5	5 nines	5 nines	6 nines
10	5 nines	5 nines	6 nines
15	5 nines	6 nines	6 nines

Table 4
Analysis of the system availability (expressed as number of nines) as function of the number of super-peers.

Number of super-peers	Group availability
1	5 nines
2	7 nines
3	7 nines

7. Related work

Peer clustering is a well known approach for segmenting a data dissemination infrastructure into a number of groups. Such approach has been the subject of extensive research in many different fields with respect to heterogeneous objectives. Table 5 provides a taxonomy of current related work that applied clustering in several IT research communities, pointing-out the objectives addressed by each work.

7.1. Peer-to-peer networks

The most known application of peer clustering is on peer-to-peer infrastructures by means of so-called *Super-Peer Networks*, such as FastTrack [50] (*i.e.*, the protocol used by Kazaa, Grokster, iMesh, and Morpheus file sharing programs). Super-Peer Networks are architected by means of two different types of peers: normal peers, where user applications are running, and super peers, which have duties of control and management. Normal peers are connected to a single super-peer, so to form the so-called peer cluster, while super-peers are connected among each other through a proper overlay. From the outside world, super-peers act as representatives of their assigned cluster and receive all the requests from other clusters (*i.e.*, they hold metadata over the data of connected normal peers so to reply to queries coming from other clusters). Within a cluster, super-peers allow normal peers to be interconnected with all other peers (both belonging to the same cluster, but also participating to other clusters), and they manage joining and leaving procedures. Super-Peer Networks have been introduced to resolve the inefficiencies that the pure Peer-to-peer approach exhibits in terms of fast and effective research of data, bottlenecks caused by the resources heterogeneity of participating peers, and scale at the Internet level [60].

Since the scale of such networks has been increasing, how super-peers are interconnected becomes a key aspect to be properly investigated. In the classic Super-Peer

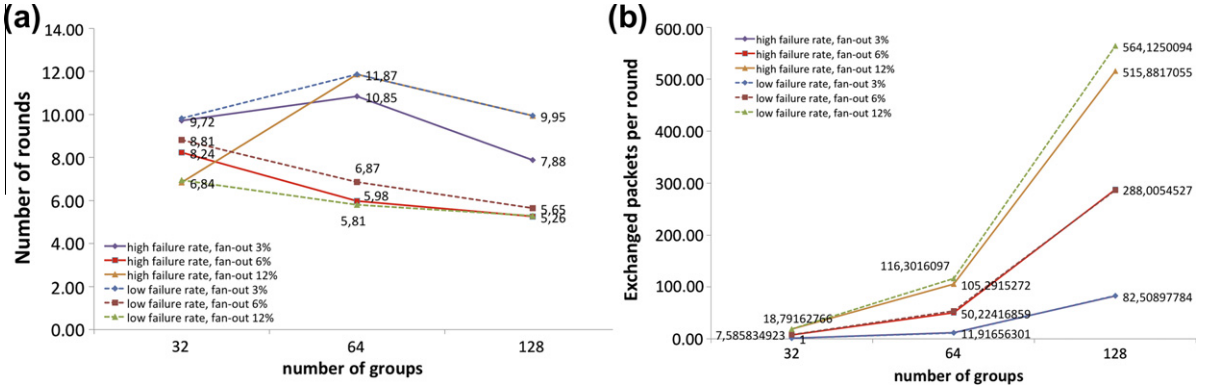

Fig. 13. Experimental results, (a) Average number of round to deliver a notification; (b) average number of messages exchanged during a round. Dashed lines report results obtained with a low failure rate.

Table 5
Taxonomy of related works in clustering.

Research communities	Solution	Objective
Peer-to-peer networks	Flat Super-Peer Networks [50]	Scalability
	Hierarchical Super-Peer Networks [51]	Scalability
	Topology-aware hierarchical Super-Peer Networks [52]	Scalability and efficiency
Multicast services	NICE [53]	Scalability
	TOMA [54]	Scalability
	Topology-aware clust-ered multicast [55]	Scalability and efficiency
	OMNI [56]	Scalability and real-time
	Quilt [57]	Scalability
	Hierarchical gossiping [58]	Scalability and reliability
Publish/subscribe middleware	Interest clustering [59]	Scalability and efficiency

Networks, the overlay interconnecting super-peers is architected in a classic pure peer-to-peer way. However, such approach has been questioned when the number of super-peers is high. Therefore, several works, such as [51], have proposed a hierarchical organization of super-peers, *i.e.*, super-peers are clustered in groups, each one with super-peers of second level, then the latter ones are further grouped in clusters of third level and so on until there is only one peer at the top of the hierarchy and clustering of higher level cannot be performed. In addition, to lower the performance overhead related to a hierarchical interconnection of super-peers, network-awareness has been introduced so to realize an efficient topology based on proximity information [52]. Specifically, only close peers are clustered during each iteration of the clustering process.

7.2. Multicast services

Based on the experience from efficient construction of peer-to-peer networks, clustering has also been applied within the context of multicast services, mainly for scalability reasons. In fact, researchers have observed that clustering is able to alleviate the issue of building and maintaining multicast trees, to optimize the resource usage by reducing the required control overhead, to provide lower and more stable performances in extreme large multicast infrastructures. NICE [53] has been the first, and widely known, Application Level Multicast (ALM) solution to use clustering: it applies clustering within a multicast group, *i.e.*, the set of nodes interested to receive a certain message, and adopts the hierarchical interconnection of super-peers as mentioned for [51] in peer-to-peer networks. TOMA, proposed in [54], is slightly different than NICE: it does not have a hierarchy for interconnecting super-peers, but a backbone made of proxies properly deployed by the service provider for conveying multicast traffic respecting service level agreements established with users. Also with respect to ALM, we witness the same evolution of clustering by including topology-related data to improve efficiency of the clustered architecture [55].

The usage of clustering in multicast services is not only limited to the enhancement of scalability properties, but it also aims to improve other aspects of multicast services. There are several concrete examples. For instance, OMNI [56] exploits a clustered organization of its nodes within an overlay to benefit real-time applications. Specifically, the overlay consists of two different types of nodes: multicast service nodes (MSN) form the backbone of the overlay and provides efficient data dissemination to meet the requirements of real-time applications running on end-host nodes. The overlay among MSNs is autonomously organized with the intent of reducing latency. An other interesting application of clustering is presented in [57], where Quilt uses clustering for resolving the deployment and scalability issues that characterize IP Multicast. In fact, the main problem of IP Multicast is that it is not supported all over Internet, especially in its backbone. A narrowed routing domain managed by a single organization, such as a LAN, typically provides a support to IP Multicast traffic; however, outside such domains routers do not convey IP Multicast traffic. Quilt applies the well-known technique of tunneling in combination with clustering to glue together IP Multicast-enabled "islands": a routing domain where IP Multicast is enabled is considered as a cluster, while an overlay, the authors used OMNI ALM, is used to perform inter-cluster communication.

Clustering has been also used to address the issue of jointly providing scalability and reliability in Internet-scale multicast services. The main flaws of approaches for guaranteeing data delivery over faulty Internet-scale networking infrastructures is that they can not scale up when incrementing the number of destinations or the amount of published data, and the lack of adaptivity to the experienced network conditions. For example, gossip-based dissemination suffers of the generation of a large number of repairing messages, which may overwhelm routers and exacerbate the faulty behavior of the network. Specifically, gossip imposes the same traffic load all over the infrastructure, even in portions where the network behavior is less faulty and demands lower intervention by the gossip protocol to achieve successful deliveries. Clustering has been used in [58] for dealing with these problems: nodes are organized in a hierarchical manner, called Leaf Box Hierarchy, similarly as seen for NICE, and the gossip protocol is made adaptive by adjusting its parameters with respect to the position of a node within the hierarchy. Specifically, the targets of a gossip round are chosen among the neighbors within the hierarchy so to reduce network traffic by limiting gossiping between nodes too far away. The usage of acknowledged messages allows calibrating the gossip protocol to the real needs of the network: if all messages are acknowledged, gossiping is never triggered, while when losses are experienced then

gossip rounds are issued. Since gossip is network-aware, this assures that additional traffic is experienced only where needed, so to alleviate the flaws that occur when trying to have reliability in Internet-scale multicast services.

7.3. Publish/subscribe middleware

The solutions that we have discussed with respect to multicast services can also be considered as topic-based publish/subscribe middleware. In addition, within this research community another innovative application of clustering has been proposed, such as clustering interest, instead of nodes. In publish/subscribe services the definition of destinations is not network-centric, *i.e.*, by specifying their addresses as happens in UDP protocol or the address of a multicast group as occurs in IP Multicast, but it is data-centric, *i.e.*, by indicating the interest in the type or content of events that the destinations are willing to receive. Therefore, interest becomes another dimension for the scale of a publish/subscribe service that has to be opportunely managed: when expressions of interest increase too much, the overhead to manage them can compromise the efficiency of the publish/subscribe service. A brilliant solution to keep the service efficient even if there is a large number of such expressions is interest clustering, *i.e.*, nodes that exhibit the same interests are placed closer to each other on the ALM. This concept is similar to subscription regionalism, *i.e.*, nodes that are geographically close also share the same interests, which has been observed in some particular application scenarios, such as news spreading of particular topics (politics or sport) that are of interest for a certain country but not for another one. In [59], there are several examples where interest clustering is applied for improving scalability.

7.4. Progress beyond the state of the art

The work presented in this paper goes beyond the state of the art described in the previous subsection by using Peer clustering not only for scalability (as seen in [59] or [51]), but also for reliability improvement. In fact, it has been applied also for improving the provided QoS so to better tailor the applied reliability means to the network conditions. In addition, it goes beyond what was done in [58], since the presented approach can be used to make different reliability approaches (not only gossiping) coexisting within the context of the overall data dissemination infrastructure.

8. Final remarks

This paper proposed TODAI, a novel approach for data dissemination in LCCIs, based on a two-layered super-peer organization, the semi-active replication of super-peers, and the epidemic delivery of messages among peer groups. The approach is shown to be able to scale up to thousands of nodes with desirable availability and resiliency properties. In particular, the use of peer groups and the adoption of the semi-active replication scheme for super-peers allow obtaining a 7 nines availability for each individual group. In addition, the proactive nature of the proposed epidemic dissemination approach delivers a resiliency, which is an increasing function of the number of nodes involved in the dissemination task. Compared to the standard solution based on RTPS, our approach is able to deliver up to 5 nines of resiliency for 128 groups (8192 peers), while keeping a delivery latency of 87 ms, against the 3 nines resiliency and 62 ms latency achievable with RTPS. Finally, the proposed approach does not suffer of sending buffer overflow problems, which on the other side limits the size of systems where RTPS can be successfully adopted. Hence, the proposed dissemination approach is promising to build scalable and resilient DDS-based systems over Internet-federated large-scale critical infrastructures.

The adopted evaluation approach, based on Stochastic Activity Networks, can be easily exploited and extended to conduct further performance and dependability assessment campaigns in the future, *i.e.*, to evaluate the resiliency of TODAI with respect to weaker assumptions or against a wider set of threats, including security attacks, which are a serious concern in LCCIs. As an example, it is possible to assume a less controlled environment for the internal system domain, which could expose peer nodes belonging to the same group to network failures. As for security attacks, while there is still a strong debate in the community, a concrete example is [61], about the difficulty/impossibility to model the behavior of attackers (and hence to assume realistic values for attack probability and attack success probability), it could be interesting to assess the impact of well-known attacks on the behavior of the system. Examples are Distributed Denial of Service attacks affecting a set of super-peers at the same time, or attacks causing super-peers behaving in an erratic way, such as, randomly loosing/duplicating messages, or introducing fake messages in the overlay.

Acknowledgements

This work has been partially supported by the Italian Ministry for Education, University, and Research (MIUR) in the framework of the Project of National Research Interest (PRIN) "DOTS-LCCI: Dependable Off-The-Shelf based middleware systems for Large-scale Complex Critical Infrastructures".

References

[1] C. Esposito, D. Cotroneo, A. Gokhale, D.C. Schmidt, Architectural evolution of monitor and control systems – issues and challenges, Introduction paper for the Special Issue on Data Dissemination for Large scale Complex Critical Infrastructures at International Journal of Network Protocols and Algorithms 2 (3) (2010) 1–17.
[2] P. Marwedel, Embedded System Design, Springer, 2006.
[3] SESAR, 2011. <http://www.eurocontrol.int/sesar/-public/subsite_homepage/home-page.html>.
[4] NASPI, 2011. <http://www.naspi.org>.
[5] PENS, 2011. <http://www.eurocontrol.int/communications/public/standard_page/pens.html>.
[6] J.P.G. Sterbenz, D. Hutchison, E.K. Çetinkaya, A. Jabbar, J.P. Rohrer, M. Schöller, P. Smith, Resilience and survivability in communication networks: strategies, principles, and survey of disciplines, Computer Networks: Special Issue on Resilient and Survivable Networks 54 (8) (2010) 1245–1265.

[7] P.Th. Eugster, P.A. Felber, R. Guerraoui, A. Kermarrec, The many faces of publish/subscribe, ACM Computing Surveys 35 (2) (2003) 114–131.
[8] OMG. Data Distribution Service (DDS) for Real-Time Systems, v1.2, 2007. <http://www.omg.org>.
[9] C. Esposito. Data Distribution Service (DDS) Limitations for Data Dissemination w.r.t. Large-scale Complex Critical Infrastructures (LCCI), 2011. <http://www.mobilab.unina.it>.
[10] C. Diot, B.N. Levine, B. Lyles, H. Kassan, D. Balendiefen, Deployment numbers for the IP Multicast services and architecture, IEEE Networks – Special Number Multicasting 14 (1) (2000) 78–88.
[11] Y. Chu, S.G. Rao, S. Seshan, H. Zhang, A case for end system multicast, IEEE Journal on Selected Areas in Communications (JSAC) 20 (8) (2002) 1456–1471.
[12] J.F. Meyer, Performability modeling of distributed real-time systems, Computer Performance and Reliability (1983) 361–372.
[13] C. Esposito, C. Di Martino, M. Cinque, D. Cotroneo, Effective data dissemination for large-scale complex critical infrastructures, in: Proceedings of the 3rd International Conference on Dependability (DEPEND 2010), 2010, pp. 64–69.
[14] W. Vogels, R. van Renesse, K. Birman, Using epidemics techniques for building ultra-scalable reliable communication systems, in: Proceedings of the Workshop on New Visions for Large-Scale Networks (LSN): Research and Applications, 2001.
[15] O. Ozkasap, End-to-end epidemic multicast loss recovery: analysis of scalability and robustness, Computer Communication 32 (4) (2009) 668–678.
[16] W.H. Sanders, J.F. Meyer, Stochastic Activity Networks: Formal Definitions and Concepts. Lecture Notes in Computer Science, Springer, Berlin, 2001. 2090/2001:315–343.
[17] L. Northrop et al. Ultra-large-scale Systems, The Software Challenge of the Future, 2006. <http://www.sei.cmu.edu/uls/>.
[18] G. Muhl, L. Fiege, P. Pietzuch, Distributed Event-Based Systems, Springer, 2006.
[19] R. Meier, V. Cahill, Taxonomy of distributed event-based programming systems, The Computer Journal 28 (5) (2002) 602–626.
[20] S.P. Mahambre, M. Kumar, U. Bellur, A taxonomy of QoS-aware, adaptive event-dissemination middleware, IEEE Internet Computing 11 (4) (2007) 35–44.
[21] OMG, DDS Interoperability Protocol (DDSI), v2.1, 2009. <http://www.omg.org>.
[22] S. Lin, D. Costello, M. Miller, Automatic-repeat-request error-control schemes, IEEE Communications Magazine 22 (12) (1984) 5–17.
[23] S.R. Chandran, A selective repeat ARQ scheme for point-to-multipoint communications and its throughput analysis, ACM SIGCOMM Computer Communication Review 16 (3) (1986) 292–301.
[24] M. Xiong, J. Parsons, J. Edmondson, H. Nguyen, D.C. Schmidt, Evaluating technologies for tactical information management in net-centric systems, in: Proceedings of SPIE: Defense Transformation and Net-Centric Systems, vol. 6578, 2007, pp. 657–668.
[25] A. Corsaro, The Data Distribution Service for Real-Time Systems, 2010. <http://www.drdobbs.com/architecture-and-design/222900238>.
[26] K.C. Almeroth, The evolution of multicast: from the mbone to interdomain multicast to internet2 deployment, IEEE Network 14 (1) (2000) 10–20.
[27] J.-H. Cui, J. Kim, D. Maggiorini, K. Boussetta, M. Gerla, Aggregated multicast – a comparative study, Cluster Computing 8 (1) (2005) 15–26.
[28] N. Bonmariage, G. Leduc, A survey of optimal network congestion control for unicast and multicast transmission, Computer Networks: The International Journal of Computer and Telecommunications Networking 50 (3) (2006) 448–468.
[29] J.F. De Rezende, S. Fdida, Scalability issues on reliable multicast protocol, in: Proceedings of COST 237 Workshop, 1999.
[30] N. Feamster, D.G. Andersen, H. Balakrishnan, M.F. Kaashoek, Measuring the effects of internet path faults on reactive routing, ACM SIGMETRICS Performance Evaluation Review 31 (1) (2003) 126–137.
[31] A. Markopoulou, F. Tobagi, M. Karam, Loss and delay measurements of internet backbones, Computer Communications 29 (10) (2003) 1590–1604.
[32] J. Seibert, D. Zage, S. Fahmy, C. Nita-Rotaru, Experimental comparison of peer-to-peer streaming overlays: an application perspective, in: Proceedings of the 33rd IEEE Conference on Local Computer Networks, 2008, pp. 20–27.
[33] M. Wiesmann, F. Pedone, A. Schiper, B. Kemme, G. Alonso, Understanding replication in databases and distributed systems, in: Proceedings of the 20th International Conference on Distributed Computing Systems (ICDCS 2000), 2000, pp. 464–474.
[34] H. Garcia-Molina, Elections in a distributed computing system, IEEE Transactions on Computers (TC) C-31 (1) (1982) 48–50.
[35] P.Th. Eugster, R. Guerraoui, S.B. Handurukande, P. Kouznetsov, A.-M. Kermarrec, Lightweight probabilistic broadcast, ACM Transaction on Computer Systems 21 (4) (2003) 341–374.
[36] P. Costa, M. Migliavacca, G.P. Picco, G. Cugola, Introducing reliability in content-based publish-subscribe through epidemic algorithms, in: Proceedings of the 2nd international workshop on Distributed event-based systems (DEBS 03), 2003, pp. 1–8.
[37] D. Deavours, G. Clark, T. Courtney, D. Daly, S. Derisavi, J.M. Doyle, W.H. Sanders, P.G. Webster, The mobius framework and its implementation, IEEE Transactions on Software Engineering 28 (10) (2002) 956–969.
[38] S. Chiaradonna, F. Di Giandomenico, P. Lollini, Evaluation of Critical Infrastructures: Challenges and Viable Approaches. Architecting Dependable Systems V, Lecture Notes in Computer Science, Springer, Berlin/ Heidelberg, 2008. 5135/2008:52–77.
[39] A. Bondavalli, M. Nelli, L. Simoncini, G. Mongardi, Hierarchical modelling of complex control systems: dependability analysis of a railway interlocking, Journal of Computer Systems Science and Engineering 16 (4) (2001) 249–261.
[40] A. Bondavalli, S. Chiaradonna, D. Cotroneo, L. Romano, Effective fault treatment for improving the dependability of COTS and legacy-based applications, IEEE Transactions on Dependable and Secure Computing 1 (4) (2003) 1545–5971.
[41] D. Cotroneo, C. Di Martino, Field data based modeling of sender based message logging protocols for supercomputers checkpointing, 2010, pp. 294–301.
[42] M. Cinque, D. Cotroneo, C. Di Martino, Automated generation of performance and dependability models for the assessment of wireless sensor networks, IEEE Transactions on Computers 2011;99(PrePrints).
[43] K.K. Goswami, R.K. Iyer, Simulation of software behavior under hardware faults, in: Proceedings of the Twenty-Third International Symposium on Fault-Tolerant Computing (FTCS-23), 1993, pp. 218–227.
[44] K.I. Ravishankar, T. Dong, Experimental analysis of computer system dependability. Fault-tolerant computer system design, Prentice-Hall, Inc., 1996. 282–392.
[45] R. Mullen, The lognormal distribution of software failure rates: origin and evidence, in: Proceedings of the International Symposium on Software Reliability Engineering (ISSRE 98), 1998, pp. 124–133.
[46] G. Hasslinger, O. Hohlfeld, The Gilbert–Elliott model for packet loss in real time services on the internet, in: Proceedings of the 14th GI/ITG Conference on Measuring, Modelling and Evaluation of Computer and Communication Systems, 2008, pp. 1–15.
[47] A. Bavier, M. Bowman, D. Culler, B. Chun, S. Karlin, S. Muir, L. Peterson, T. Roscoe, T. Spalink, M. Wawrzoniak, Operating system support for planetary-scale network services, in: Proceedings of the First Symposium on Networked Systems Design and Implementation (NSDI 04), 2004, pp. 253–266.
[48] A. Corsaro, CoFlight eFDP. available on the DDS portal at portalsomgorg/dds/sites/default/files/dds_06-09-05_0pdf 2006;.
[49] E. Halepovic, R. Deters, B. Traversat. Performance Evaluation of JXTA Rendezvous. Lecture Notes in Computer Science: On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE – Springer Berlin/ Heidelberg 2004;3291/2004:1125–1142.
[50] J. Liang, R. Kumar, K.W. Ross, The fasttrack overlay: a measurement study, Computer Networks 50 (6) (2006) 842–858.
[51] K.M. Hammouda, M.S. Kamel, Hierarchically distributed peer-to-peer document clustering and cluster summarization, IEEE Transactions on Knowledge and Data Engineering 21 (5) (2009) 681–698.
[52] E.K. Lua, X. Zhou, Network-aware superpeers-peers geometric overlay network, in: Proceedings of the 16th International Conference on Computer Communications and Networks (ICCCN 2007), 2007, pp. 141–148.
[53] S. Banerjee, S. Lee, B. Bhattacharjee, C. Kommareddy, Scalable application layer multicast, in: ACM SIGCOMM Computer Communication Review – Proceedings of the 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM 2002), vol. 32, issue 4, 2002, pp. 205–217.
[54] L. Lao, J.-H. Cui, M. Gerla, S. Cheng, A scalable overlay multicast architecture for large-scale applications, IEEE Transactions on Parallel and Distributed Systems 18 (4) (2007) 449–459.

[55] X. Zhanga, X. Lia, W. Luoa, B. Yan, An application layer multicast approach based on topology-aware clustering, Computer Communications 32 (6) (2009) 1095–1103.
[56] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, S. Khuller, OMNI: an efficient overlay multicast infrastructure for real-time applications, Computer Networks 50 (6) (2006) 826–841.
[57] Q. Huang, Y. Vigfusson, K. Birman, H. Li, Quilt: a patchwork of multicast regions, in: Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems (DEBS 2010), 2010, pp. 184–195.
[58] I. Gupta, A.-M. Kermarrec, A.J. Ganesh, Efficient and adaptive epidemic-style protocols for reliable and scalable multicast, IEEE Transactions on Parallel and Distributed Systems 17 (7) (2006) 593–605.
[59] L. Querzoni, Interest clustering techniques for efficient event routing in large-scale settings, in: Proceedings of the second international conference on Distributed event-based systems (DEBS 2008), 2008, pp. 13–22.
[60] B. Yang, H. Garcia-Molina, Designing a super-peer network, in: Proceedings of the 19th International Conference on Data Engineering (ICDE 2003), 2003, pp. 49–60.
[61] D.M. Nicol, W.H. Sanders, K.S. Trivedi, Model-based evaluation: from dependability to security, IEEE Transactions on Dependable and Secure Computing 1 (1) (2004) 48–65.

Marcello Cinque graduated with honours from University of Naples, Italy, in 2003, where he received the PhD degree in computer engineering in 2006. Currently, he is Assistant Professor at the Department of Computer and Systems Engineering (DIS) of the University of Naples Federico II. Cinque is chair and/or TPC member of several technical conferences and workshops on dependable, mobile, and pervasive systems, including IEEE PIMRC, DEPEND, and ACM ICPS. His research interests include dependability analysis of mobile and sensor systems, and middleware solutions for mobile ubiquitous systems.

Catello Di Martino received the MS degree with honours and PhD in computer engineering from the University of Naples in 2006 and 2009, respectively. Currently, he is a research fellow at the Computer Engineering and Systems Department (DIS) at the University of Naples. His interests include dependability assessment techniques of WSN and large scale computer systems. Di Martino has been at the Center for Reliable and High-Performance Computing of the University of Illinois, Urbana Champaign for 18 months doing research with R.K. Iyer. He has also worked as a consultant for Critical Software Inc. in the context of verification and validation campaigns of large satellite projects.

Christian Esposito graduated in Computer Engineering at Universitá di Napoli Federico II in 2006, and got his PhD at the same university in 2009. Currently he is a post-doc researcher at Consorzio Inter-Universitario Nazionale per l'Informatica (CINI) and Department of Computer and Systems Engineering (DIS) at Universitá di Napoli Federico II. His main interests include positioning systems for mobile ad hoc networks, benchmarking aspects of publish/subscribe services, and reliability strategies for data dissemination in large-scale critical systems. He is currently involved in the EU project called CRITICAL STEP (FP7-PEOPLE-2008-IAPP-230672), and italian project called PRIN DOTS-LCCI (PRIN-2008-LWRBHF).