Reliable and Timely Event Notification for Publish/Subscribe Services over the Internet

Christian Esposito, Marco Platania, and Roberto Beraldi

***Abstract*—The publish/subscribe paradigm is gaining attention for the development of several applications in Wide Area Networks (WANs), due to its intrinsic time, space and synchronization decoupling properties that meet the scalability and asynchrony requirements of those applications. However, while the communication in a WAN may be affected by the unpredictable behavior of the network, with messages that can be dropped or delayed, existing publish/subscribe solutions pay just a little attention to addressing these issues. On the contrary, applications such as business intelligence, critical infrastructures, and financial services require delivery guarantees with strict temporal deadlines.**

In this paper, we propose a framework that enforces both reliability and timeliness for publish/subscribe services over WAN. Specifically, we combine two different approaches: gossiping, to retrieve missing packets in case of incomplete information, and network coding, to reduce the number of retransmissions and, consequently, the latency. We provide an analytical model that describes the information recovery capabilities of our algorithm, and a simulation-based study, taking into account a real workload from the Air Traffic Control domain, that evidences how the proposed solution is able to ensure reliable event notification over a WAN within a reasonable bounded time window.

***Index Terms*—Publish/Subscribe, Reliability, Timeliness, Network Coding, Gossip**

I. Introduction

In the last few years, we have been facing the development of an increasing number of large scale applications such as online gaming, messaging, social networking, and business intelligence. In addition, we have see the development of federations in Large-scale Complex Critical Infrastructures (LCCIs) of independent critical systems previously designed for "closed" Local Area Networks (LANs), such as Air Traffic Control (ATC) systems, financial infrastructure monitoring, and maritime surveillance. These applications are typically characterized by a large number of participants scattered across the world that communicate by exchanging messages on a Wide Area Network (WAN), such as the Internet.

Manuscript received xx XXX, 200X; revised xx XXX, 200X.

The first author is currently affiliated to the Institute of High Performance Computing and Networking (ICAR) National Research Council (CNR), Napoli 80131 (Italy). This work has been conducted when he was affiliated to Dipartimento di Informatica e Sistemistica (DIS), Universitá Federico II, Napoli 80125 (Italy). The second author is currently affiliated to Johns Hopkins University, Dept. of Computer Science, Baltimore, MD 21218, USA. This work was conducted when he was affiliated to Dipartimento di Ingegneria Informatica Automatica e Gestionale "A. Ruberti", Sapienza University of Rome, 00185 Roma (Italy). The last author is affiliated to the Dipartimento di Ingegneria Informatica Automatica e Gestionale "A. Ruberti", Sapienza University of Rome.

Christian Esposito is the corresponding author (for the e-mail address see http://wpage.unina.it/christian.esposito/Contacts.htm).

The publish/subscribe paradigm is an appealing solution as messaging middleware because it offers the time, space and synchronization decoupling properties [1] that distributed applications require. However, while this kind of middleware fits the generic asynchrony and scalability requirements of large scale applications, it completely or partially lacks the support of Quality of Service (QoS) guarantees. Existing solutions, in fact, either provide only a best-effort service [2], [3], or address just a single requirement at a time (e.g., message ordering [4], bounded delivery time [5], or reliable delivery [6]). While some applications can rely on a best effort service (messaging, social networking), other applications may require several non-functional requirements to be met. As a concrete example, let us consider the new ATC framework under development within the context of the SESAR EU Project [7], whose goal is to ensure the safety and fluidity of air transportation, and to reduce the costs of air traffic management by switching to off-the-shelf computing equipments and moving some of the services to the public Internet. The ATC scenario is characterized by several geographically sparse actors, such as air traffic controllers, weather stations, airport staff, and pilots, each of them being both producer (publisher) and consumer (subscriber) of information that compose the flight plan of a specific aircraft *en route*. In this context, a message loss or a late delivery can compromise the mission of the overall system, leading to negative consequences in terms of efficiency, economic losses, consumer dissatisfaction and even indirect harm to people. On the one hand, since WANs are affected by data losses [8], [9] that can compromise the correctness of the ATC system, the communication infrastructure must implement a technique to tolerate these losses (i.e., published events must be *reliably* delivered to all interested subscribers). On the other hand, published data must be delivered within a known time bound [10] (i.e., information delivery must respect a *timeliness* constraint), so that the system can promptly react to any sudden change of flight route.

The aim of this paper is the design of a framework that ensures reliable and timely event notification in a WAN and can be plugged in on top of a generic publish/subscribe system. Our approach combines two different techniques, each known to be reliable and timely, respectively: (i) gossip [11], a distributed retransmission protocol, and (ii) network coding [12], a Forward Error Correction (FEC) scheme. We consider a scenario in which a publisher publishes events (i.e., application level messages) and redundant information on UDP overlay links built over the Internet, which exhibits a non negligible probability to have burst losses [8]. Then, the intended subscribers apply a gossip strategy to recover from possible

lost data. We provide a theoretical model to evaluate the potential benefits of gossip to retrieve the missing information. In addition, we describe a simulation-based study conducted on a real workload taken from the previously described ATC scenario, which evidences how the use of coding to send redundant information is able to reduce the latency and the message overhead for a reliable event delivery.

Differently from other solutions, our approach addresses reliability and timeliness at the same time. These two constraints, in fact, have been typically considered as separate aspects, sometimes resulting in conflict between each other, making current solutions not suitable in the presence of QoS-demanding systems. A naive solution to achieve reliability is to use TCP connections. However, the UDP protocol is more suitable for delay-sensitive applications, because it provides a minimized transmission delay by avoiding the connection setup process and allowing a designer to implement his/her own flow control and retransmission schemes.

Finally, let us remark that the use of IP QoS mechanisms and architectures, such as IntServ and DiffServ, is not sufficient to solve the problem that we want to address. On the one hand, the use of these solutions, as well as dedicated links, imposes a high cost on a service provider, while one of the main reasons why several applications are moving to the public Internet is due to cost reduction. On the other hand, a QoS mechanism, such as DiffServ, lacks granularity in data traffic policy, which may lead to a degradation of the QoS for all data flows in the same class even if only one data flow generates excessive traffic. In addition, DiffServ operates on the per-IP domain and the per-hop basis: Internet Service Providers, in fact, typically do not provide a service with QoS guarantees.

The remainder of the paper is organized as follows: Section II introduces gossip and network coding. Section III discusses several solutions similar to ours within the context of the available literature on publish/subscribe services. Section IV describes the system and network model, while Section V introduces the proposed solution. The theoretical model and simulation results are provided in Section VI and Section VII, respectively. Finally, Section VIII concludes the paper with some final remarks and plans for future work.

II. Background

A. Network Coding

Random linear network coding [12] is a technique that allows us to convey the information content of n original packets, $x_1, x_2, ...x_n$, as a set of n linearly independent combinations (encoded packets) and to easily generate redundancy virtually for any lost packets by means of linear combinations of original data. Each linear combination is given by

$$y_i = \sum_{i=1}^{n} c_i x_i$$

where coefficients c_i are taken uniformly at random over the set $0, ..., q-1$, with the all zero coefficients case excluded. All operations are performed over the Galois Field $GF(2^w)$, with $2^w = q$. Each coded packet y_i is equipped with the coefficients c_i used to produce that packet and that will be used by destinations in the decoding phase. Note that the overhead due to the generation of the coefficients is very modest, as it is equal to sending nw additional bits (for example, for n = 10 and q = 256 this means just 80 bits; considering a typical packet size of 1KB, the overhead is less than 0.1%).

The beneficial effect of network coding is twofold. On the one hand it is well known that network coding can achieve the mincut bound for a single multicast source on a directed graph [12], [13]. This allows to exploit transmission bandwidth for recovery operations. On the other hand, it reduces the average number of gossip rounds needed to retrieve m missed packets from $O(m\, log(m))$ to $O(m)$ [14]. Overall, network coding lays the foundations for an efficient and fast recovery where redundancy is judiciously added to the dissemination or recovery operations. In fact, in [15] we have previously shown that the probability that a plain received packet is *useful* to reconstruct the whole event increases linearly with the number of missed packets m, while under a coding scheme this probability increases exponentially.

B. Gossiping

The gossip paradigm [11] is based on the so-called *epidemic approach*, where an event is disseminated like the spread of a contagious disease or the diffusion of a rumor. Specifically, a node stores a received message in a buffer of size b, and forwards it a limited number of times t, named *fan-in* to a randomly-selected set of nodes of size f, called *fan-out*. Many variants of gossip algorithms exist

1) *push*: messages are forwarded to the other nodes as soon as they are received;
2) *pull*: nodes periodically send to other nodes a set of recently-received message identifiers. If a missing message is detected by comparing the received set with the local history, then a transmission is requested;
3) *push/pull*: a node forwards only the identifier of the last received message. If one of the receivers does not have such a message, then it makes an explicit pull request.

Gossip-based protocols have several advantages that have been thoroughly studied: a few initial infection points are sufficient to quickly infect the whole population as the number of infected nodes grows with an exponential trend. Moreover, these algorithms are also strongly resilient to the premature departure of several nodes, making them very robust against failures. The gossip approach has been successfully applied to a variety of application domains, such as database replication, cooperative attack detection, resource monitoring, and publish/subscribe-based data dissemination. Taking into account the properties of an ideal event dissemination service, most of such algorithms based on the gossip paradigm are able to deliver a huge amount of events in a geographically distributed setting with nice reliability properties.

III. Related work

Although many real world applications require support to chieve QoS, the majority of current research prototypes [3], [2], [16] operate on a best-effort basis [17]. Hermes [18] shows reliability properties just from the fault-tolerance point

of view: it uses techniques to enable event brokers to recover after a failure, but it does not provide support for client-specific delivery requirements as a service guarantee. On the contrary, PADRES [6] additionally tolerates message losses and guarantees publication delivery: overlay path redundancy is exploited both for the routing process and fast information recovery. Reliability can also be ensured in JEDI [4] by means of TCP connections; however, it is well-known that the use of TCP in multicast tree overlays exhibits a low throughput [19] in practical applications. IndiQoS [5] focuses on respecting timeliness in event dissemination. Subscribers specify the latency constraint as an attribute of their subscriptions. Brokers are responsible for reserving a path from subscribers to publishers based on the aggregate traffic they manage, by means of a constant number of deterministic attempts in order to find routes that satisfy those requirements. Finally, note that several QoS properties (including message ordering, timeliness, and reliability) are ensured by some publish/subscribe commercial systems, such as Tibco [20], DDS [21] and JMS [22]. However, these systems are designed to properly work in small/medium LANs, while their performance poorly degrades over WANs.

Differently from the previous papers, our goal is to propose a framework for publish/subscribe systems deployed in WAN that ensures reliability and timeliness at the same time, by exploiting gossiping and network coding. One of the first solutions in combining these two approaches is [23], where the combination is performed at the receiver side: coding is used only when a sender has to retransmit in push mode the received data. We considerably differ from this work by: (i) applying coding at the publisher side, (ii) investigating several gossip strategies and the improvement that coding can bring to them, and (iii) evaluating the effects of coding not only on the delivery latency but also on the imposed overhead and in different network conditions. Another similar work is presented in [24], which differs from ours due to its theoretical nature (i.e., dissemination approaches are only studied by means of analytical models), and its evaluation metrics (i.e., gossiping with and without coding are studied only with respect to latency). In addition, [24] assumes a plain data dissemination only performed by means of push- or pull-based gossip rounds. The authors in [25] propose a combination of FEC and retransmission schemes to limit, with a high probability, the packet loss rate of overlay channels to a target value q. Specifically, the protocol restricts the number of retransmissions to at most one, in order to minimize the end-to-end latency.

The main difference with our solution lies in the fact that the protocol presented in [25] is suitable for streaming applications, where a minimum packet loss rate can be tolerated, while we concentrate on more generic and demanding systems, where reliable delivery is of paramount importance. Finally, in [15] we have described the benefit of using network coding combined with gossip, and in [14] we have improved this description with a theoretical model and a simulation-based study. Starting from these contributions, we complement the theoretical model in [14] with an analysis of the recovery capability of gossip algorithms in a tree-based overlay network. In addition, we extend the simulation-based study of [14] to assess the power of network coding during data dissemination; in [14] network coding was applied only when retrieving missing information.

IV. System and network model

A. Node architecture

Each node in the system implements the architecture depicted in Fig. 1. It is composed by three building blocks: application, *Reliability and Timeliness* layer and a publish/subscribe Event Notification Service (ENS).

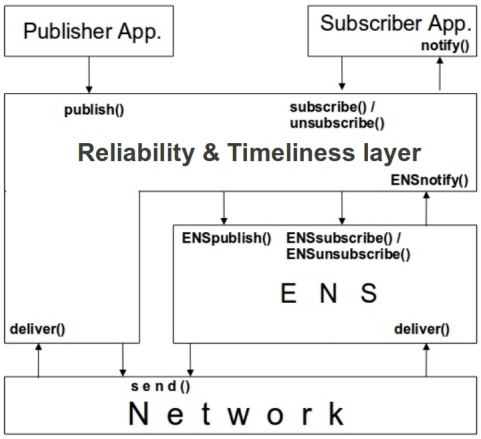


Fig. 1. Architecture implemented by all system nodes.

Application: The architecture we designed is targeted for several applications deployed over a WAN, for example maritime surveillance, air traffic control, collaborative security, next generation intelligence platform, stock market, etc. We individuate two distinct roles: information producers, i.e. *publishers*, and information consumers, i.e. *subscribers*.

Producers of information can be sensors that capture data from an environment (i.e., temperature, humidity, enlightenment); systems or devices, for example firewalls that produce log files for monitoring applications; radars or stock market sites that generate periodic updates; data sources that update database in a cloud. On the contrary, consumers of information are systems that infer environmental conditions (i.e., fires, flooding) by analyzing data detected by sensors; trading applications that buy and sell actions based on data produced by stock market sites; flight processors that analyze information about flight plans.

Applications can be both publishers and subscribers. As an example, processing engines in complex event detection applications can be subscribers of raw events generated by firewalls of the monitored system, and at the same time, they can also be publishers of complex events obtained by correlating raw events coming from different sources.

Reliability and Timeliness layer: The *Reliability and Timeliness* layer wraps the adopted ENS in order to ensure reliability and timeliness properties in the event dissemination as defined below:

- *Reliable delivery*: each published event is delivered to all its intended destinations.

- *Timely delivery*: Let t be the time when an event is published and t' the time when it was delivered to a subscriber. Any delivered event is such that $t' - t < \Delta$

This layer intercepts events published by the application or notified by the ENS, executes several operations to enforce the two requirements, and then it publishes the events on the ENS or notifies them to the subscriber application. The *Reliability and Timeliness* layer may also require additional interaction among nodes, and it can be done by accessing a point-to-point communication primitive that can be offered by the operating system or by other solution like an overlay network. In addition, it is worth noticing that the presented *Reliability and Timeliness* layer exposes the same interface of the ENS; thus neither the applications, nor the ENS must be changed in order to work with our framework.

Event Notification Service: The communication model used in our architecture follows the event-based paradigm [1], with exchanged information that takes the form of *event*. The interaction among publishers and subscribers is mediated by a distributed *Event Notification Service (ENS)*, that implements the following interface:
publish(): invoked to publish events in the system;
subscribe(): invoked by subscribers to declare an interest in a topic or content;
unsubscribe(): invoked by subscribers to unsubscribe from a previously declared interest;
notify(): invoked by the ENS to deliver events to subscribers according to their interests.

B. Network model

We consider system nodes connected through an overlay network built on top of Internet links. Several works [8], [9] in literature show that communications over the Internet can be affected by losses, spanning from a single to several packets being dropped. Therefore, we assume that links among nodes are not reliable, and exhibit a loss pattern characterized by *Packet Loss Rate* (PLR), which is the probability to lose a packet, and *Average Burst Length* (ABL), which is the mean number of consecutive lost packets. In particular, the adopted network model is the *Gilbert-Elliott* [26], one of the most-commonly applied in performance evaluation studies, due to its analytical simplicity and the good results provided in practical applications on wired IP networks [27], [28]. The adopted Gilbert-Elliott model is a first order Markov chain with two states: a "Good" state, where packets are not lost, and "Bad" or lossy state.
The adopted network model is characterized by four transition probabilities:

- the probability P to pass from the "Good" state to the "Bad" one;
- the probability $1 - P$ to remain in the "Good" state;
- the probability Q to pass from the "Bad" state to the "Good" one;
- the probability $1 - Q$ to remain in the "Bad" state.

P and Q are related to PLR and ABL as follows [29]:

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

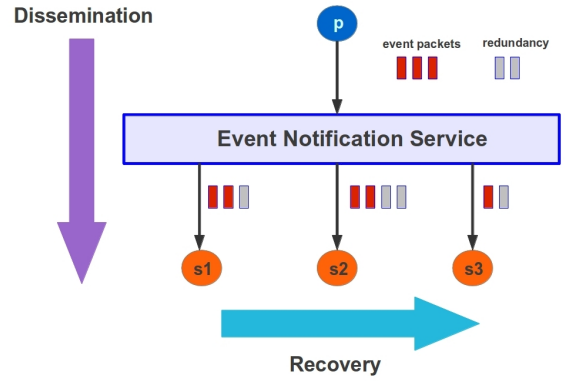


Fig. 2. **The protocol in practise. *Dissemination*: a publisher p publishes an event and redundant packets on the ENS. *Recovery*: based on the specific gossip strategy in use, a subscriber can recover missing packets. In the figure above, s1 and s2 have a sufficient number of packets to reconstruct the original event, while s3 needs a gossip round to gather the missing information.**

V. Proposed protocol

We consider the *Reliability and Timeliness* layer as a composition of two independent blocks: (i) one implements a network coding protocol that aims to reduce the delivery time of an event over the entire set of subscribers, and (ii) one implements a gossip-based algorithm to recover from possible event losses. To this end, we assume that each node can access a peer sampling primitive [30], [31] to obtain a sample (i.e., another subscriber) to gossip with.

The protocol we propose to reliably and timely deliver events is composed by two phases, as depicted in Fig. 2:
Dissemination: a publisher divides an event into n plain packets and publishes them over the ENS. Moreover, a additional packets are also added to the published event. We call a as the *redundancy* of the protocol. We separately consider two different cases: (i) *no-coding:* the a packets are randomly selected among the n plain packets; (ii) *coding:* the a packets are linear combinations of the n packets;
Recovery: the subscribers use a gossip protocol to gather possible lost events. This phase strictly depends on the gossip style in use: if the push or push/pull strategy is enabled, then, a node that notifies an event (i.e., it has received enough packets to reconstruct the whole event) disseminates to the other f nodes the received packets or the event identifier respectively. When the pull style is enabled, periodically a node disseminates to other f nodes the identifiers of the last notified events.

A consideration needs to be done: the delay introduced by encoding and decoding operations is low for two reasons: (i) on the encoding side, the operations take not so much time because they are simple linear combinations and several libraries can be used to perform them efficiently; (ii) on the decoding side, operations are progressive, i.e., they are based on a *decoding matrix* that is built concurrently with the reception of packets. The decoding matrix is maintained in the

triangular form by using Gaussian elimination [12]. Each time a new encoded packet arrives, it is inserted into the matrix only if the new carried coefficients increase the rank of that matrix. Therefore, the decoding delay overlaps with the transmission delay. Further details about the delay penalty introduced by the coding operations can be found in [12].

In the following, before describing the algorithm in detail, we introduce the local data structure maintained by publishers and subscribers.

Local data structure to each publisher p_i: each publisher locally maintains the following data structures:

- id_e: a unique identifier associated to an event e produced by p_i.
- $packets$: a set variable, initially empty, that contains all packets in which an event is fragmented.
- $redundancy$: a set variable, initially empty, that contains redundant packets: they can be plain or coded packets.
- $coding$: a variable that indicates if coding is enabled.
- $publishedEvents$: a set variable, initially empty, that contains the couple $\{e, redundancy\}$.

Local data structure to each subscriber s_i: each subscriber maintains locally the following data structures:

- $coding$: a variable that indicates if coding is enabled.
- $incomingPackets_{id_e}$: a set variable, initially empty, that contains all packets received for the event with identifier id_e.
- $notifiedEvents$: a set variable, initially empty, that contains the tuple $\{e, e_{id}, t\}$, where t represents the gossip $fan-in$.
- $lastNotifiedEvents$: a set variable, initially empty, that contains the identifiers of the recently notified events and it is used during a pull-based gossip recovery procedure.
- $gossip\_mode$: a variable that indicates the gossip strategy used in the recovery phase of the algorithm.
- $contacts$: a set variable that contains the identifiers of the nodes returned by the peer sampling service.

a) PUBLISH()**:** The algorithm for a PUBLISH() operation is reported in Fig. 3. To simplify the pseudocode, we defined the following basic functions:

- fragment(e): it fragments an event e in n plain packets.
- encode($packets, a$): it implements the coding operation by generating a linear combinations of the n plain packets contained in $packets$.
- selectPacket($packets, a$): it randomly selects a plain packets among the n contained in $packets$.

The PUBLISH() operation works as follows: the event e is fragmented in n plain packets and stored in the $packets$ data structure (line 01). Then, a redundant packets are generated (lines 02-05): depending on the variable $coding$ in line 02, these packets can be linear combinations of the original n packets (line 03), or random packets selected among the original ones (line 04). The $n+a$ packets are published on the ENS (lines 06-11) and, then, stored in the $publishedEvents$ data structure (line 12).

Note that each published packet is also provided with the identifier id_e and a *boolean* value that indicates if coding is enabled. The n original packets are always published without coding (07).

```
operation PUBLISH(e):

(01) packets ← FRAGMENT(e);
(02) if (coding = TRUE)
(03)   then redundancy ←encode(packets, a);
(04)   else redundancy ←selectPacket(packets, a);
(05) endif
(06) for each (pkt ∈ packets);
(07)     ENSpublish (< pkt, FALSE, id_e >);
(08) endfor
(09) for each (red ∈ redundancy);
(10)     ENSpublish (< red, coding, id_e >);
(11) endfor
(12) publishedEvents ← publishedEvents ∪ {e, redundancy};
(13) packets ← {};
```

Fig. 3. The publish() protocol for a publisher p_i.

b) NOTIFY()**:** The algorithm for a NOTIFY() operation is reported in Fig. 4 and Fig. 5. To simplify the pseudocode, we defined the following basic functions:

- decode(pkt, id_e): it implements the decoding process by maintaining a triangular matrix for packets related to the event with identifier id_e. It returns a decoded packet.
- canReconstructEvent($incomingPackets_{id_e}$): it is a $boolean$ function that checks if the received packets are enough to fully reconstruct the event with identifier id_e. If so, the function returns $TRUE$, otherwise $FALSE$.
- reconstructEvent($incomingPackets_{id_e}$): it actually reconstructs an event e with identifier id_e from the packets contained in the $incomingPackets_{id_e}$ data structure.
- getPeer(f): it provides access to the peer sampling service by returning f random subscribers currently in the system. f represents the $fan-out$ of the gossip algorithm.
- pushEvent(e, id_e, s_j): this function starts a push-based gossip procedure by sending to a subscriber s_j the received event e.
- pushEventId (id_e, s_j): this function starts a push/pull-based gossip procedure by sending to a subscriber s_j the identifier id_e of the received event e.
- sendRecentHistory($lastNotifiedEvents, s_j$): this function starts a pull-based gossip procedure by sending to a subscriber s_j the identifiers of the last received events contained in the $lastNotifiedEvents$ data structure.

The NOTIFY() operation works as follows: upon receiving a packet, the handlePacket function is called (line 01). This function implements the core activity for each received packet; we decided to separate it from the NOTIFY() operation in order to reuse HANDLEPACKET also for processing the incoming retransmitted packets during the recovery phase of the protocol. Depending on the kind of received packet (plain or coded, line 02), it can be simply added to the set of the incoming packets (line 03) or it requires a decoding process first (line 04). At each received packet, the algorithm checks if there are enough packets to reconstruct the event (line 07). If so, the reconstructEvent function actually reconstructs that event (line 08). Note that receiving a number of packets equal to or higher than n is a necessary but not sufficient condition

```
upon ENSNOTIFY(< pkt, coding, e_id >):

(01) handlePacket(pkt, coding, e_id);
-------------------------------------------------------------
function HANDLEPACKET(pkt, coding, e_id):

(02) if (coding = FALSE)
(03)    then incomingPackets_{id_e} ← incomingPackets_{id_e} ∪ {pkt};
(04)    else incomingPackets_{id_e} ← incomingPackets_{id_e} ∪
            {decode(pkt, id_e)};
(05) endif
(06) if (|incomingPackets_{id_e}| ≥ n);
(07)    then if (canReconstructEvent(incomingPackets_{id_e}) = TRUE)
(08)         then e = reconstructEvent(incomingPackets_{id_e})
(09)              trigger notify (e);
(10)              notifiedEvents ← notifiedEvents ∪ {e, id_e, t};
(11)              if (gossip_mode = PUSH ∨ PUSHPULL)
(12)                then contacts ← getPeer(f);
(13)              endif
(14)              if (gossip_mode = PUSH)
(15)                then for each (s_j ∈ contacts)
(16)                         pushEvent(e, id_e, s_j);
(17)                     endfor
(18)                else if (gossip_mode = PUSHPULL)
(19)                then for each (s_j ∈ contacts);
(20)                         pushEventId (id_e, s_j);
(21)                     endfor
(22)              endif
(23)         endif
(24)    endif
```

Fig. 4. The notify() protocol for a subscriber s_i.

```
upon TIMEOUT():

(01) for each (< e, id_e, t > ∈ notifiedEvents)
(02)     lastNotifiedEvents = lastNotifiedEvents ∪ {id_e};
(03)     t = t − 1;
(04)     notifiedEvents ← notifiedEvents / {< e, id_e, t >};
(05)     if (t > 0)
(06)       then notifiedEvents ← notifiedEvents ∪ {< e, id_e, t >};
(07)     endif
(08) endfor
(09) contacts ← getPeer(f);
(10) for each (c_j ∈ contacts)
(11)     sendRecentHistory(lastNotifiedEvents, c_j);
(12) endfor
(13) lastNotifiedEvents ← {};
```

Fig. 5. The expiration of the timeout fires a new pull-based gossip execution.

to fully reconstruct an event. In fact, without coding it is required to receive n *different* packets, while with coding n *independent* linear combinations are needed. When an event is fully reconstructed, the subscriber triggers the notify operation (line 09) and inserts it in the set of received events (line 10). Then, if the push or push/pull gossip strategy is enabled, the subscriber asks to the peer sampling service a set of f random nodes currently in the system (lines 11-13), and sends them the received event (lines 14-17 for the push-based style) or its identifier (lines 18-21 for the push/pull-based style).

Fig. 5 shows the pseudocode for a pull-based gossip strategy. Periodically a subscriber generates a set with the identifiers of the last received events (line 02) and sends it to f random nodes by means of the sendRecentHistory function (lines 09-12). The number of times that an event identifier can be sent to other subscribers is regulated by the parameter t, i.e., the $fan-in$ of the algorithm. Each time an identifier is inserted in the $lastNotifiedEvents$ data structure, its value of t is decreased by one (line 03). When $t = 0$, the tuple related to that identifier is no more updated in the $notifiedEvents$ data structure (lines 04-06).

VI. Analytical model

In this Section we describe a mathematical model to grasp the recovery ability that gossip brings to our protocol. We derive a success probability for event delivery as the probability to receive the information after the dissemination and gossip phases. We also illustrate the benefit of using network coding to reduce the number of retransmissions, by considering a pull gossip strategy with fan-out fixed to 1. In Section VII we extend this analysis with a more detailed simulation-based study that takes into account a more realistic scenario, several gossip strategies and a varying fan-out value.

A. Assumption

We assume the ENS implemented as a tree-based overlay network, with publisher and subscriber roles implemented by the same nodes that constitute the ENS. Specifically, we consider a single publisher, i.e., the root of the tree, while other nodes play the role of subscriber. To make the analysis simpler, we consider a regular complete tree of depth L and connectivity D, having $D^{L+1}-1$ nodes. This assumption will be relaxed in Section VII, where we consider a generic tree-based overlay network.

We call *rank* of a node the rank of its decoding matrix, i.e., the dimension of the subspace spanned by the linear combinations received by that node. In addition, we consider that an event is fragmented in n packets. Then, we make the following further assumptions:

- A1: the rank of a node holding r coded packets is $min\{r, n\}$, i.e., all packets are linearly independent from each other;
- A2: the rank of the union of two decoding matrixes with rank r_1 and r_2 is $min\{\{r_1 + r_2\}, n\}$;
- A3: nodes are connected through independent Gilbert-Elliot channels. At the beginning of each round, the channel state is at the steady state, i.e., it is in the bad state with probability PLR;
- A4: gossip is synchronous and organized in consecutive global rounds, during which all nodes execute a gossip operation. The effect of a gossip operation is visible only at the end of the round. In addition, a node always contacts a node belonging to another subtree. This assumption will be weakened later;
- A5: the protocol described in the previous Section considers a recovery strategy in which a subscriber contacts other subscribers at random. As such, in the theoretical analysis we assume that the root of the tree cannot be contacted during a gossip interaction.

Our analysis focuses on how the rank of a generic node at level h varies over time. We will denote with $\pi_{hi}^{(k)}$ the probability that at the end of round k the rank of a node at level h is i. Hence, $\pi_{hn}^{(k)}$ is the probability that a level h node detects the event at round k or earlier.

B. Dissemination phase

The initial dissemination process of packets along the tree is characterized by the following $(n+1)\times(n+1)$ probability

matrix

$$P_D = \begin{bmatrix} 1 & 0 & \dots & 0 & \dots & 0 \\ P_{10} & P_{11} & \dots & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ P_{i0} & P_{i1} & \dots & P_{ii} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ P_{n0} & P_{n1} & \dots & \dots & \dots & P_{nn} \end{bmatrix}$$

This is a triangular and stochastic matrix where the element P_{ij} is a probability describing the relationship between two adjacent nodes. Specifically, P_{ij} is the probability that the rank of a node at level h is j, given that the rank of its ancestor node at level $h-1$, which is sending the packets to it, is i.

The transmission of packets occurs in L steps. During each step, nodes at level h send packets to nodes at level $h+1$, using independent channels, so that L steps are required for the information to reach the leaves. The probability distribution of the source node, occupying level 0, is trivially characterized by the following $(n+1)$ probability vector

$$\pi_0^{(0)} = [0, 0, \dots, 1]$$

indicating that the rank of the source is n. The probability distribution

$$\pi_h^{(0)} = [\pi_{h0}^{(0)}, \pi_{h1}^{(0)}, \dots, \pi_{hn}^{(0)}]$$

of nodes at level h ($h = 1, ..., L$) is obtained from the following iterative equation

$$\pi_h^{(0)} = \pi_{h-1}^{(0)} P_D = \pi_0^{(0)} P_D^h \tag{1}$$

The probability P_{ij} is computed starting from the probability that m out of n packets pass through a Gilbert-Elliot (GE) channel, denoted as $P_T(n, m)$, see [32] for its derivation[1].

Considering that for sending i packets nodes in general add a_i redundant packets, it is easy to see that for $j < i$

$$P_{ij} = P_T(i + a_i, j)$$

i.e., for the rank of the destination node to be j, the node should receive only j out of the $i + a_i$ sent packets. Finally,

$$P_{ii} = 1 - \sum_{j<i} P_{ij}$$

The probability that a node at level h gets the full event content is P_{nn}^h. In addition, P_{nn} denotes the probability of an event passing through the GE channel, i.e., the probability that no more than a_n packets are erased by the channel. Accordingly, P_{nn} can be rewritten as

$$P_{nn} = \sum_{d=0}^{a_n} P_T(n + a_n, d) \tag{2}$$

[1]The paper reports the probability $P_L(n, d)$ that d out of n packets are lost; clearly, $P_T(n, m)$ can straightforwardly be derived from $P_L(n, d)$.

C. Gossip phase

The analysis of the gossip phase follows a similar approach. The k-th gossip round is characterized by the following $(n+1) \times (n+1)$ *Gossip* matrix $P_G(k)$

$$P_G(k) = \begin{bmatrix} P_{00}(k) & P_{01}(k) & \dots & \dots & \dots & P_{0n}(k) \\ 0 & P_{11}(k) & \dots & \dots & \dots & P_{1n}(k) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & P_{ii}(k) & \dots & P_{in}(k) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & \dots & 1 \end{bmatrix}$$

where $P_{ij}(k)$ is the probability that after round k, the rank of the gossiping node increases from i to j. As the rank of node cannot decrease after a gossip round, $P_G(k)$ is a stochastic and triangular matrix.

We can compactly describe the initial probability distribution of nodes' rank at the beginning of the gossip process through the following $(L+1) \times (n+1)$ matrix

$$\Pi^{(0)} = \begin{bmatrix} \pi_0^{(0)} \\ \pi_1^{(0)} \\ \vdots \\ \pi_L^{(0)} \end{bmatrix} = \begin{bmatrix} \pi_{00}^{(0)} & \pi_{01}^{(0)} & \dots & \pi_{0n}^{(0)} \\ \pi_{10}^{(0)} & \pi_{11}^{(0)} & \dots & \pi_{1n}^{(0)} \\ \vdots & \vdots & \dots & \vdots \\ \pi_{L0}^{(0)} & \pi_{L1}^{(0)} & \dots & \pi_{Ln}^{(0)} \end{bmatrix}$$

where $\pi_{hi}^{(0)}$ is the probability that at time 0 the rank of a node at level h is i, see Equation 1.

The probability $\pi_{hj}^{(k+1)}$ that at the end of round $k+1$ the rank of a node at level h is j is clearly given by

$$\pi_{hj}^{(k+1)} = \sum_{i \le j} \pi_{hi}^{(k)} P_{ij}(k)$$

which is expressed in matrix form as

$$\Pi^{(k+1)} = \Pi^{(k)} P_G(k)$$

In order to compute $P_{ij}(k)$ it is worth introducing $PH_r(k)$, defined as the probability that the rank of the contacted node is r. We first observe that due to Assumptions A4 and A5, the total number of contactable nodes is indeed $D^L - 1$ (we recall that the root cannot be contacted), whereas the number of contactable nodes at level $h > 0$ is $D^h - D^{h-1}$; hence, the level of the contacted node is h with probability

$$\frac{D^h - D^{h-1}}{D^L - 1} = \frac{(D-1)}{D^L - 1} D^{h-1}$$

Now, for the contacted node to have rank r, the following events should occur

- the level of the contacted node is h, this happens with probability $\frac{(D-1)}{D^L-1} D^{h-1}$;
- the rank of such a node is r, this happens with probability π_{hr}^{k-1}.

Hence, for $r = 0, .., n$

$$PH_r(k) = \frac{D-1}{D^L - 1} \sum_{h=1}^{L} D^{h-1} \pi_{hr}^{(k-1)}$$

Let i be the rank of the contacting node, say A, and suppose that the rank of the contacted node, say B, is r. The rank of A passes from i to $j < n$, if the following events occur:

- the rank of B is $r, j \leq i + r < n$, B sends $r + a_r$ linear combinations to A, but $a_r + r - (j - i)$ packets get lost;
- the rank of B is $r, i + r \geq n$, B sends $(n - i) + a_{n-i}$ linear combinations to A, but $a_{n-i} + n - j$ packets get lost.

Hence, for $j < n$

$$P_{ij}(k) = \sum_{r=j-i}^{n-i-1} PH_r(k) P_L(r + a_r, a_r + r - (j - i)) +$$

$$\sum_{r=n-i}^{n} PH_r(k) P_L(n - i + a_{n-i}, a_{n-i} + n - j)$$

whereas

$$P_{in}(k) = 1 - \sum_{i \leq j < n} P_{ij}(k)$$

Finally, $P_{nn}(k) = 1$.

D. Introducing blindness

So far, we have assumed that the selected node B does not belong to the node's A subtree. This maximizes the effectiveness of a recovery operation as the two nodes experience independent loss patterns. In a pure gossip protocol, however, the selection is blind; thus, A and B may belong to the same subtree thus reducing the recovery capability. For example, if B is a child of A, then the first gossip round is not useful at all, simply because B cannot have received more information than A. On the other hand, this can be no longer true for subsequent rounds, as B may have recovered from some node C on a different subtree.

In order to macroscopically capture this aspect, we introduce in the previous model an 'effectiveness' factor α, which modifies the transition probability of the recovery phase from i to j. We can think at this parameter in this way: a recovery operation is fully effective with probability α, and not useful with probability $1 - \alpha$. The transition probabilities for the gossip phase are modified as follows

$$P'_{ij} = \alpha P_{ij}, \quad j > i$$

while $P'_{ii} = 1 + \alpha(P_{ii} - 1)$. For $\alpha = 0$, all gossip rounds are not useful, i.e., the recovery capability is null, whereas $\alpha = 1$ means maximum theoretically possible recovery capability.

E. Performance metric

A key performance metric of the protocol is the ability of nodes to decode an event. In a balanced tree of degree D, the total number of receivers is

$$\frac{D^{L+1} - D}{D - 1}$$

The success probability of a node at level h at time k corresponds to $\pi_{hn}^{(k)}$, whereas the probability that a node in the network gets the event is obtained as the ratio between the average number of detected events by the total number of nodes

$$P_{succ}(k) \approx \frac{D - 1}{D^{L+1} - D} \sum_{1 \leq h \leq L} \pi_{hn}^{(k)} D^h$$

For $k = 0$ it is possible to derive a closed form for the success probability. At the end of the first phase, in fact, the average number of decoded events, i.e., events that are fully received by nodes, is

$$\sum_{h=1}^{L} D^h p^h = \sum_{h=1}^{L} (Dp)^h$$

where $p = P_{nn}$ is the probability that at least n out of $n + a$ packets pass through a GE channel (see Equation 2).

By dividing this number by the total number of nodes we get [2]

$$P_{succ}(0) = \frac{(Dp)^{L+1} - Dp}{D^{L+1} - D} \times \frac{D - 1}{Dp - 1} \tag{3}$$

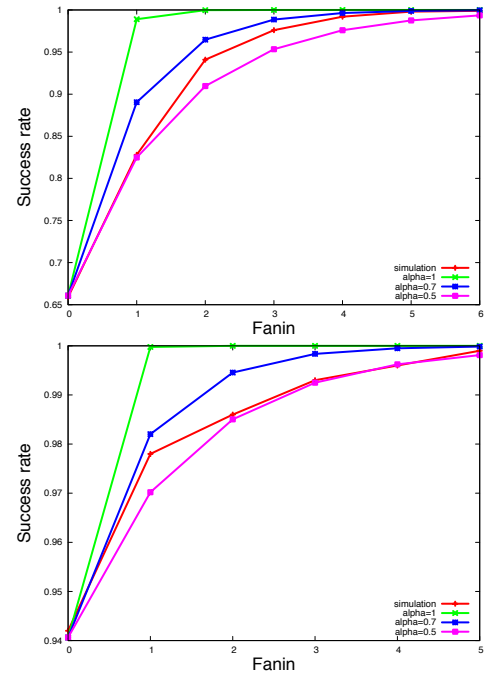


Fig. 6. Success rate vs fan-in. $PLR = 2\%, ABL = 2, n = 16$. No coding (top) and coding with redundancy = 3 (bottom). For clarity of presentation, a different scale has been used on the y-axis of the two plots.

F. Model validation

In this Section we report some numerical result obtained through the *numpy* library for Python. We consider a balanced binary tree with 15 nodes, $ABL = 2$ and $PLR = 2\%$. Fig. 6 compares the success probability P_{succ} predicted by our theoretical model and the one estimated through a simulation study (see Section VII). The no coding (top) and coding with redundancy 3 (bottom) cases are reported. The Figure shows how coding strengthens the recovery ability of gossip (note

[2]For $p = D^{-1}$ the success probability is $\frac{L(D-1)}{D^{L+1}-D}$.

that for clarity of presentation we have used a different scale on the y-axis of the two plots).

In Fig. 6, the simulated result is well predicted when $\alpha = 0.5$ (the result provides a lower bound), while $\alpha = 0.7$ provides a good upper bound. It is worth noting that in a binary tree a node has half of the chances to select a node in the other subtree. Hence, when $\alpha = 0.5$ we are conservatively assuming that recovery with nodes in the same subtrees has no effect at all. In general, $\alpha = \frac{D-1}{D}$ provides a lower bound. Hence, the model becomes more and more accurate for fat tree, i.e., when D is large. In addition, $\alpha = 1$ provides the ideal result: any uninformed gossip algorithm, i.e., that is not aware about the contacted node's content in advance, cannot do better than this.

G. Effect of the channel characteristics

Fig. 7 shows the effect of the channel's parameters PLR (top) and ABL (bottom) on the success rate (see Equation 3) of the dissemination phase of the protocol. The success rate decreases with PLR, with the effect being contrasted when additional redundancy is applied. The effect of ABL is more interesting to analyze. If PLR is constant, an increase in ABL will modify the loss pattern in a way that a burst of losses is followed by a no-loss burst of longer duration. Losses are thus more likely to hit packets of a same event, leaving window of time during which more events can pass through, without any of their packets being erased by the channel.

For $ABL = 1$, a single redundant packet is able to neutralize the effect of a single loss inside an event. The success probability increases of a value equal to the probability that an event gets only a single packet loss (≈ 0.45). However, if the ABL increases to 2, an event may undergo to more than a loss and the neutralization effect of the single redundant packet diminishes. Higher ABL will however create longer periods of no losses, and this will increase the success rate. The figure also shows how two redundant packets will protect the event completely when $ABL = 1$. An increase in ABL can now only decrease the success rate.

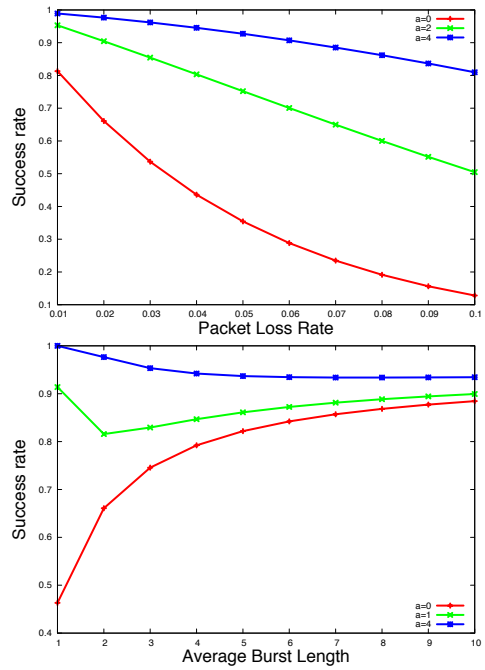


Fig. 7. Effects of the channel characteristics on the success rate of the dissemination phase of the protocol (no recovery). Effect of the Packet Loss Rate with $ABL = 2$ (top) and effect of the Average Burst Length with $PLR = 2\%$ (bottom) are reported.

VII. Evaluation

In this Section, we report the results of the conducted simulation analysis, by considering push, push/pull and pull gossip-based recovery strategies. We also relax some of the assumptions made in the previous Section, i.e., we consider a generic overlay network and asynchronous gossip interactions (assumption A4). We have implemented our solution on OMNET++ and used the SCRIBE implementation of the OVERSIM library as the ENS. The workload we have used in the simulations has been taken from the Co-Flight system, a real deployment of the new ATC framework realized in the SESAR EU project: (i) 1 publication per second, and (ii) 40 nodes in the system (estimated number of ATM entities involved in the first Co-Flight deployment). All nodes play both the publisher and subscriber roles.

The network parameters, instead, are set as follows: communication delay = 50 msec, PLR = 2%, ABL = 2. These values represent an approximation of the worst case results collected in a previous study [33] on network performance conducted on PlanetLab. In addition, we have modeled the time to obtain a coded packet equal to 5 msec, while the time for the dual operation is equal to 10 ms. We have also assumed the packet size to be equal to the payload of MTU in Ethernet (i.e., 1472 bytes) so that an event is fragmented into 16 packets. We have simulated a period of 1000 publications and reported the average of different experiments on the same scenario.

The metrics we have considered in our study are the following:

Success rate: the ratio between the number of received events and the number of the published ones. It is a metric to evaluate the reliability of the proposed approach.

Loss rate: percentage of the subscribers that have not received a given event.

Fan-out for complete reliability: gossip fan-out to achieve a success rate equal to 1.

Overhead: the ratio between the total number of packets exchanged during an experiment and the number of packets generated by publishers (that is the number of published events times the number of packets in which an event is fragmented to be conveyed by the network). It is a measure of the traffic load imposed on the network, and should be kept as lower as possible in order to avoid congestions.

Latency: the time taken to disseminate an event to all subscribers. The standard deviation indicates possible performance fluctuations due to the applied fault-tolerance mechanisms, highlighting timing penalties that can compromise the timeliness requirement.

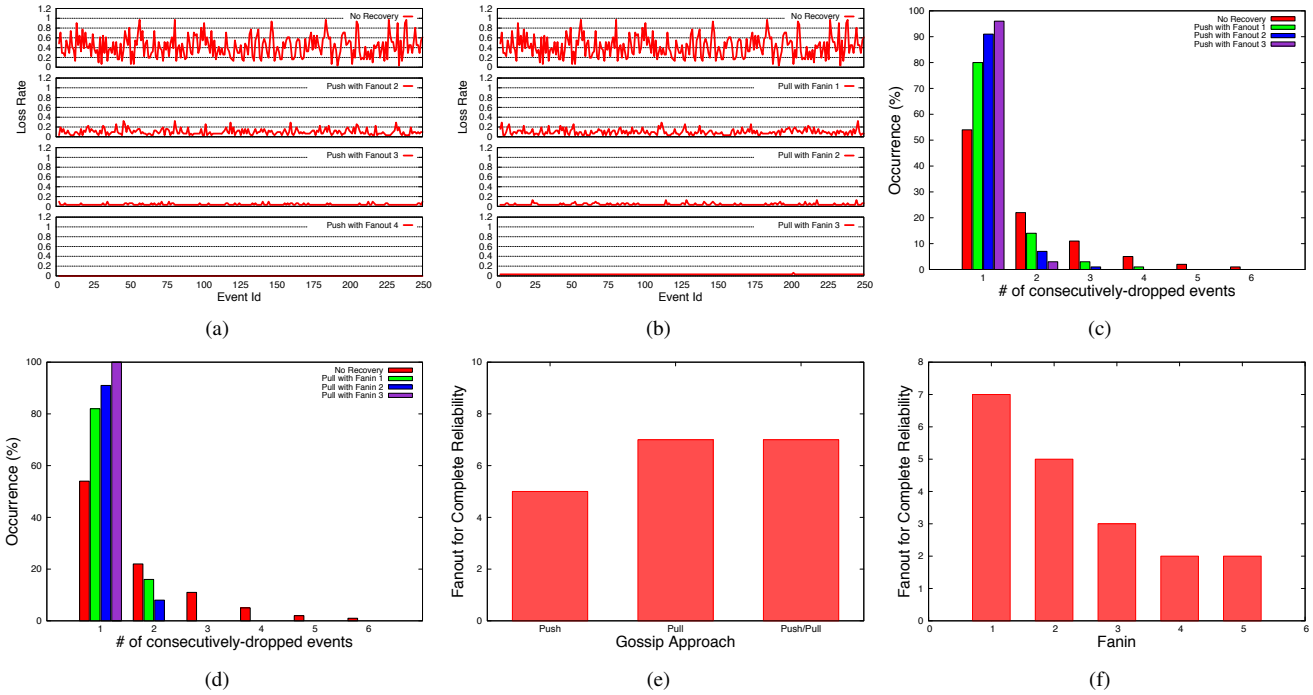


Fig. 8. **(a-b) Loss rate of single events for push and pull gossip by varying respectively fan-out and fan-in (fan-out set to 2). (c-d) Percentage of consecutive lost events in push and pull gossip. (e-f) Fan-out for a complete reliability for the three gossip strategies.**

TABLE I
LATENCY CONSTRAINTS IN THE CO-FLIGHT SYSTEM.

Category	95%	99.8%
D-1	1 sec	2 sec
D-2	2 sec	4 sec
D-3	5 sec	8 sec

Concerning the timeliness, we consider the three latency constraints defined in the context of the Interoperability Consultancy Group (ICOG) for the deployment of the Co-Flight system[3]. Table I reports the three latency categories for exchanging Flight Objects (FO), i.e., a data structure that contains all information referring to a flight, such as its current position, airport source, airport destination, etc.:
D-1: FO containing co-ordination data with the next ATC entity along the aircraft route;
D-2: FO for updating a constraint affecting to the next ATC entity along the aircraft route;
D-3: FO sent only "for information" to a distant system instance.
The second and third columns in Table I indicate that 95% and 99.8% of subscribers must receive a published event within x and y seconds, respectively. For our simulation analysis we considered the most demanding category, i.e., D-1. Hence, with respect to the timeliness definition given in Section IV, we split the parameter Δ into Δ_1 and Δ_2 such as: (i) $\Delta_1 = 1$ sec and $\Delta_2 = 2$ sec. In order to meet the timeliness requirement, our algorithm must ensure that 95% of subscribers receive an event within Δ_1 *and* 99.8% of subscribers within Δ_2.

The parameters we have varied in our analysis are (i) **Redundancy degree**, i.e., number of redundant packets sent by a publisher and/or a gossiping node in addition to the original ones; (ii) **Gossip fan-out**, (iii) **Gossip fan-in** (when not explicitly declared, we have assumed that it is set to 1), and (iv) **Pulling interval** (when not explicitly declared, we have assumed that it is set to 1.5 sec).

A. Gossip strategies evaluation without coding

In this Section we show how the network conditions affect our protocol and how to set gossip parameters to achieve a reliable delivery of all events without redundancy. Fig. 8(a) and 8(b) show the loss rate for push and pull gossip varying fan-out and fan-in respectively (in case of pull gossip the fan-out value is set to 2). In these mentioned figures we have shown a subset of 250 events, but the same results have been obtained by considering the whole set of published events, also in different runs. The Figures show that when applying gossiping schemes, the loss rate experienced by subscribers is reduced both in average and in variability. Moreover, the increase of fan-out has the effect of improving such a reduction, i.e., a higher fan-out for a push-based gossip implies an average reduction of the lost events of about 80%, and the same reduction is observed also in the standard deviation (this improvement based on a higher fan-out applies in general for every possible gossiping scheme, as shown in [14]). Fig. 8(e) shows how to set the fan-out value so that all events are reliably delivered by subscribers.

For a pull-based scheme also the fan-in affects the provided success rate. Fig. 8(b) shows that increasing fan-in reduces the average loss rate and its variability (similarly as achieved by

[3]Technical note for "ATM Scenarios for Data Distribution", Software Initiative between Selex-SI and University of Naples, whose content can not be publicly disclosed. The content of this deliverable is based on a study promoted by ICOG to the main industries in the European avionic domain (i.e., Thales, Indra and Selex-SI), and defines a common solution for ATC-to-ATC interoperability up to its architectural principles.

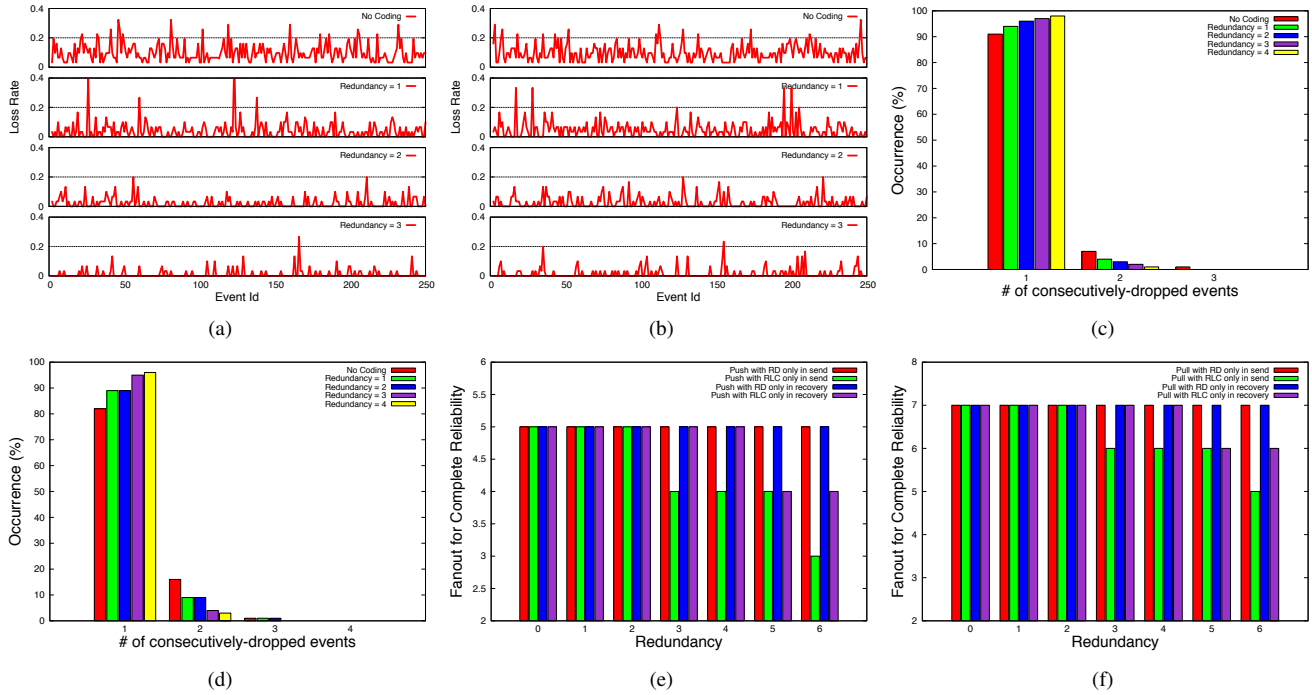


Fig. 9. **(a-b) Lost rate for single event for push and pull gossip with and without coding. (c-d) Consecutive event lost reduction augmenting redundant coded packets in push and pull gossip. (e) RLC versus RD in event dissemination and push-based recovery.**

augmenting fan-out). A direct consequence is that, as depicted in Fig. 8(f), with an increase of the fan-in the value of fan-out required for complete reliability is decreased.

In Fig. 8(c) and 8(d) we characterize how consecutive event losses are distributed in push and pull strategies, with the ABL set to 2: while during the data dissemination phase without any recovery up to 6 consecutive events can be lost, with gossip schemes most of the time we notice 1 or 2 consecutive losses. An increase of fan-out and/or fan-in value augments the probability to miss only one event rather than consecutive ones. Finally, because the push/pull strategy delivers a success rate that is close to those shown by the pull approach (as depicted in Fig. 8(e)), we do not consider push/pull in the next evaluations.

B. Gossip strategies evaluation with coding

We repeated the previous study by introducing redundant packets. In the following we refer to a coded packet as *Random Linear Combination (RLC)*, while a plain redundant packet is referred to as *Random Duplication (RD)*. In Fig. 10(a) we compare the RD and RLC coding by varying the redundancy degree. The obtained results show that RLC coding is able to improve the reliability of an event dissemination protocol without requiring a high redundancy degree. In fact, with RD a complete delivery of all events is achieved only with a redundancy equal to 29 (i.e., the event is sent almost three times), while with RLC the number of redundant packets required is more than halved. To better highlight the effectiveness of RLC against RD, let us consider Fig. 9(e) where we show the effect of these coding approaches in reducing the gossip fan-out for a reliable delivery of all events. We considered four different cases: RLC (resp. RD) applied either to the data dissemination or the push-based recovery strategy. While applying RD has no effect both in data dissemination and recovery, RLC provides a remarkable improvement especially if applied in data dissemination. This is motivated by the ability of network coding to provide *useful* packets to fully reconstruct an event (under the assumption of independent coefficients [12]), as also discussed in Section II-A. A similar result is obtained with pull gossip in Fig. 9(f). For this reason, in the following part of this paper we have considered only RLC coding.

From Fig. 9(a) to 9(d) we evaluate the event loss rate and the number of consecutive events lost for the push and the pull gossip in the presence of coding. The fan-out and fan-in values are respectively set to 2 and 1. With respect to the previous analysis, we obtain that coding causes a reduction of both the percentage of undelivered events and the number of consecutive losses.

C. Performance of the algorithm

In this Section we show the performance of the proposed protocol in terms of packet overhead and delivery latency. Fig. 10(b) compares the measured overhead for a complete reliability in push- and pull-based schemes, with and without coding. As expected, the push protocol shows the highest overhead because it is a proactive strategy: every packet is forwarded to other nodes as soon as it arrives. On the contrary, the pull protocol is a reactive approach that operates on a periodical basis: a node asks for a retransmission only when needed (i.e., a loss is detected). This obviously reduces the number of packets sent through the network. As depicted in Fig. 10(b), the use of network coding decreases the overhead generated by both strategies. This is a direct consequence of what we have previously mentioned: coding provides more *useful* packets so as to reduce the number of retransmissions.

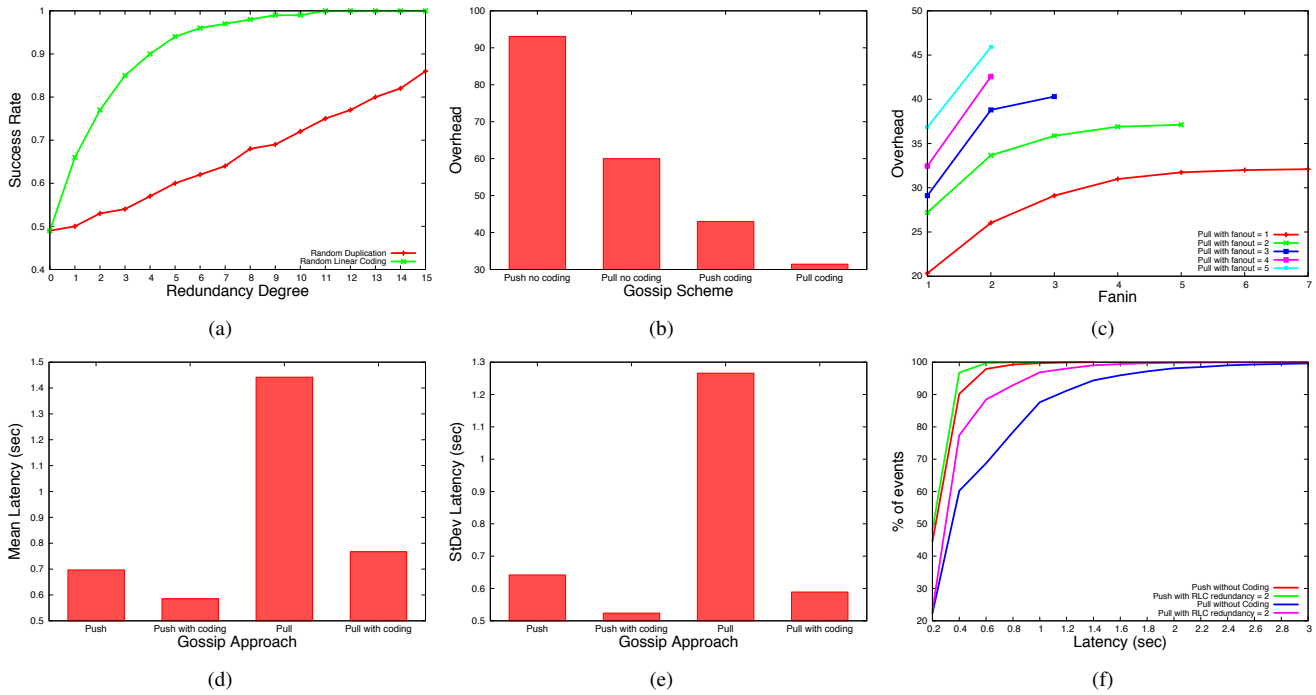


Fig. 10. **(a) Reduction of redundant packets in data dissemination with network coding. (b) Comparison of the measured overhead with and without RLC coding. (c) Variations in the measured overhead when varying fan-out and fan-in in pull gossip. (d-e) Mean value and standard deviation of latency with and without RLC. (f) CDF of the percentage of events delivered by all subscribers with time.**

Fig. 10(c) shows that reducing the fan-in in pull gossip requires a higher fan-out to achieve a reliable delivery, so that the packet overhead increases. In fact, with a higher fan-in, it is more likely that, with time, the contacted nodes have already retrieved most of the missing information; as such, the number of required retransmission decreases.

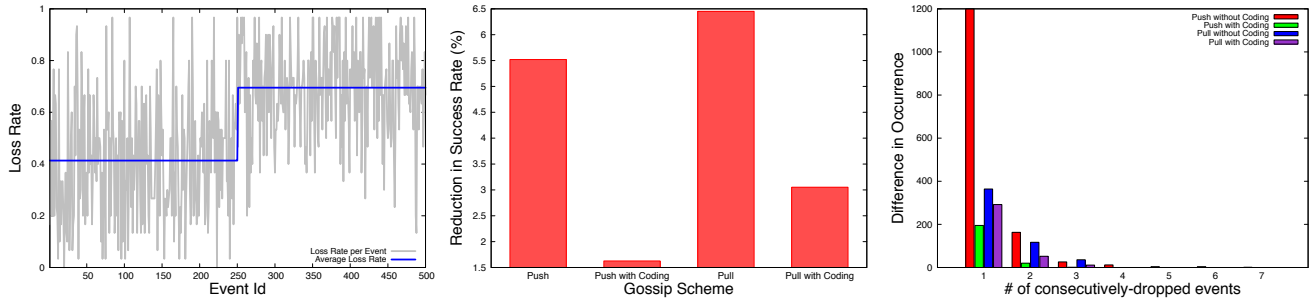
Concerning the latency, as expected the push gossip exhibits a better performance than pull (both in mean value and standard deviation, as shown in Fig. 10(d) and 10(e)), due to its periodical dissemination of the last received event identifiers. Increasing the fan-out has good effects on the mean latency and its standard deviation since this augments the probability of a node being contacted to recover lost data. We recall that a high standard deviation indicates latency fluctuations that compromise the timeliness requirement of the application. Not surprisingly, due to its ability to provide *useful* packets to recover from possible losses, and then to reduce the number of retransmissions, network coding has a good effect on the two metrics. An additional parameter for a pull-based scheme that affects its behaviour is the pulling period. In the previous results, it was set to 1.5 sec. If we reduce this value, we observe a reduction in the mean and standard deviation of the measured latency, equal respectively to 15% and 28%. Despite reducing the experienced latency, the performance of the pull-based gossip is still higher than the push-based one. On the other hand, such an increase implies a slight growth in the overhead equal to 2%. However, the reduction of the pulling interval implies a negligible variation to the average success rate and the occurrence of the loss patterns (number of consecutively-dropped events).

Finally, Fig. 10(f) shows the Cumulative Distribution Function (CDF) of the percentage of events delivered to subscribers with time. In the simulation we considered push and pull gossip without redundancy and with two redundant coded packets (the gossip parameters have been set to fan-out equal to 2, fan-in to 1 and pull interval to 0.5 sec). The push strategy is always able to satisfy the timeliness constraints defined above, even in the absence of redundancy. However, recall from the previous analysis that push gossip has the highest overhead. Pull gossip, instead, is able to satisfy the timeliness requirements only in the presence of redundant coded packets. Hence, Fig. 10(f) clearly shows that our approach is able to ensure a timely notification even in the presence of reactive approaches, that are typically subject to delay penalties.

D. Dependance on network dynamics

In this last series of simulations, we have evaluated the effects of a sudden change of network conditions on the measured delivery characteristics with the recovery strategies mentioned above. In particular, we have altered the PLR from 2% to 5%. Such a change is performed after 250 events have been published. Fig. 11(a) shows that such a sudden change implies an increase of the event loss rate (i.e., about 28%). Even if not reported, we obtained a similar result when using gossip strategies with coding (fan-out set to 2, 2 redundant packets and fan-in set to 1 and retransmission period of 0.5 sec in the case of pull). As a concrete example in terms of success rate, the push-based gossip reduces from 0.942 (when no variation occurs) to 0.890, while with coding it passes from 0.984 to 0.968.

The reduction of the number of retransmissions when using coding, evidenced also in the previous results, mitigates a possible worsening of network conditions. Specifically, Fig. 11(b) shows the reduction of the obtained success rate between the coding and no coding cases: the variation of network



(a) (b) (c)

Fig. 11. **(a) Variation of the event loss rate when PLR moves from 2% to 5%. (b) Reduction in percentage of the measured success rate due to a network conditions change. (c) Variation in the distribution of consecutive lost events when moving from PLR=2% to PLR=5%.**

conditions has a much lower impact when gossip is teamed up with coding. A similar result is obtained in Fig. 11(c), which indicates the increase of occurrences of burst losses when comparing the two cases. The PLR variation has a stronger impact on the push gossip rather than the pull one; however, in both strategies the use of coding drastically reduces the number of consecutive lost events.

E. Final considerations

We have evaluated the impact of coding through a simulation-based study conducted on a real workload taken from the ATC scenario. The obtained results demonstrate that coding helps to decrease the gossip fan-out (Fig. 9(e) and 9(f)) to achieve a reliable delivery of events, due to its ability to provide *useful* packets to recover from possible losses. This also translates into a reduction of the overhead (Fig. 10(b) and 10(c)), the mean notification latency and its standard deviation (Fig. 10(d) and 10(e)). We also defined a timeliness constraint, based on the requirements of the described ATC scenario, and evaluated in Fig. 10(f) how the proposed solution ensures the satisfaction of these constraint (i.e., 95% of subscribers must receive an event within 1 sec and 99.8% within 2 sec) We have shown that the push-based recovery strategy is able to satisfy the timeliness constraint even without coding. However, this comes at the cost of a high packet overhead. The use of coding helps to mitigate the imposed overhead, rather than reducing the notification interval. On the contrary, network coding is of paramount importance to meet the timeliness requirement when using a pull-based gossip strategy. In addition, the experienced overhead is much less than the one generated by push gossip, due to the reactive nature of pull. In this case, however, the retransmission period must be properly set to avoid delays.

VIII. Conclusions

The problem of jointly providing reliability and timeliness over a WAN is still an open issue since reliability improvements are typically obtained at the cost of severe performance fluctuations, or a stable performance is obtained by weakening the offered reliability. In this paper, we have proposed a strategy that combines coding and gossip for reliable and timely event dissemination over the Internet. We have conducted a theoretical analysis to evaluate the ability of gossip to retrieve missing information in a small number of rounds. In addition, we have evaluated the impact of coding through a simulation-based study conducted on a real workload taken from the ATC scenario. The obtained results prove that coding improves the reliability by lowering the number of nodes to be contacted during a gossip round. In addition, coding has a positive impact even on the mean notification latency, and its standard deviation, due to a decrease in the number of retransmissions required to fully reconstruct a message. As such, a decrease of latency fluctuations makes the latency itself more predictable and, in turn, helps to satisfy the timeliness constraints imposed by the applications.

In the near future, we plan to extend the contribution of this work by proposing a different mechanism to select gossip partners during the recovery phase of the protocol. In this paper we have assumed a random uniform selection mechanism; however, we can improve the efficiency of such a solution by selecting nodes with a proper heuristics. To this end, we plan to use a *polarized* gossip, in which a subscriber assigns a weight to a subset of other subscribers returned by the peer sampling service. The weight assignment follows the probability that the returned subscriber has a missing packet. Hence, a subscriber contacts f of the highest-weighted subscribers returned by the peer sampling service, i.e., those with the highest probability of having a missed packet. In particular, we plan to devise two different models for the weight assignment. One is based on the current network status, which requires us to infer the mean number of lost packets for each incoming overlay link. The second is based on the position of the nodes in the overlay network, with the idea that nodes closer to the source of information have a lower probability of having experienced a packet loss.

Acknowledgment

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”, and by the BLEND Eurostar European Project.

References

[1] P. Eugster, P. Felber, R. Guerraoui, and A.-M. Kermarrec, “The many Faces of Publish/subscribe,” *ACM Computing Surveys (CSUR)*, vol. 35, no. 2, pp. 114–131, June 2003.

[2] A. Carzaniga, D. Rosenblum, and A. Wolf, "Design and Evaluation of a Wide-Area Event Notification Service," *ACM Transactions on Computer Systems (TOCS)*, vol. 19, no. 3, pp. 332–383, August 2001.

[3] M. Castro, P. Drushel, A. Kermarrec, and A. Rowstrom, "Scribe: A Large-scale and Decentralized Application-level Multicast Infrastructure," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 20, no. 8, pp. 1489–1499, January 2004.

[4] G. Cugola, E. Di Nitto, and A. Fuggetta, "The jedi event-based infrastructure and its application to the development of the opss wfms," *IEEE Transactions on Software Engineering*, pp. 827–850, 2001.

[5] N. Carvalho, F. Araujo, and L. Rodrigues, "Scalable qos-based event routing in publish-subscribe systems," in *Network Computing and Applications, Fourth IEEE International Symposium on*. IEEE, 2005, pp. 101–108.

[6] E. Fidler, H. Jacobsen, G. Li, and S. Mankovski, "The padres distributed publish/subscribe system," in *Feature Interactions in Telecommunications and Software Systems*, vol. 8, 2005, pp. 12–30.

[7] EUROCONTROL. (2008, February) The ATM Deployment Sequence, SESAR Project Milestone Deliverable D4. [Online]. Available: www.eurocontrol.int/sesar/public/standard_page/documentation.html

[8] A. Markopoulou, F. Tobagi, and M. Karam, "Loss and Delay Measurements of Internet Backbones," *Computer Communications*, vol. 29, no. 10, pp. 1590–1604, June 2006.

[9] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, Y. Ganjali, and C. Diot, "Characterization of Failures in an Operational IP Backbone Network," *IEEE/ACM Transactions on Networking (TON)*, vol. 16, no. 4, pp. 749–762, August 2008.

[10] R. Baldoni, M. Contenti, S. Piergiovanni, and A. Virgillito, "Modeling publish/subscribe communication systems: towards a formal approach," in *Proceedings of the 8th International Workshop on Object-Oriented Real-Time Dependable Systems*. IEEE, 2003, pp. 304–311.

[11] A.-M. Kermarrec, L-Massoulié, and A. J. Ganesh, "Probabilistic Reliable Dissemination in Large-Scale Systems," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 14, no. 2, pp. 1–11, February 2003.

[12] C. Fragouli, J. L. Boudec, and J. Widmer, "Network coding: an instant primer," *Computer Communication Review*, vol. 36, no. 1, p. 63, 2006.

[13] S. Li, R. Yeung, and N. Cai, "Linear network coding," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371–381, 2003.

[14] C. Esposito, S. Russo, R. Beraldi, M. Platania, and R. Baldoni, "Achieving reliable and timely event dissemination over wan," *Proceedings of the 13rd International Conference on Distributed Computing and Networking (ICDCN)*, pp. 265–280, 2012.

[15] C. Esposito, S. Russo, R. Beraldi, and M. Platania, "On the benefit of network coding for timely and reliable event dissemination in WAN," *Proceedings of the 1th International Workshop on Network Resilience*, pp. 84–89, October 2011.

[16] L. Fiege and G. Muehl, "Rebeca event-based electronic commerce architecture, 2000."

[17] G. Muehl, L. Fiege, and P. R. Pietzuch, *Distributed event-based systems*. Springer-Verlag, 2006.

[18] P. Pietzuch and J. Bacon, "Hermes: A Distributed Event-Based Middleware Architecture," *Proceedings of 22nd International Conference on Distributed Computing Systems Workshops (ICDCSW '02)*, pp. 611–618, July 2002.

[19] K. Birman, G. Chockler, and R. van Renesse, "Toward a cloud computing research agenda," *SIGACT News*, vol. 40, no. 2, pp. 68–80, 2009.

[20] TIBCO, Inc., "Tibco rendezvous," http://www.tibco.com/products/automation/messaging/low-latency/rendezvous/default.jsp.

[21] OMG. (2007, January) Data Distribution Service (DDS) for Real-Time Systems, v1.2. [Online]. Available: www.omg.org

[22] S. Microsystems. (2002, April) Java Message Service, v1.1. [Online]. Available: docs.sun.com/app/docs/doc/816-5904-10

[23] M. Balakrishnan, K. Birman, A. Phanishayee, and S. Pleisch, "Ricochet: Lateral Error Correction for Time-Critical Multicast," *Proceedings of the 4th USENIX Symposium on Networked System Design & Implementation (NSDI 07)*, pp. 73–86, April 2007.

[24] S. Deb, M. Medard, and C. Choute, "Algebraic Gossip: A Network Coding Approach to Optimal Multiple Rumor Mongering," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2486–2507, June 2006.

[25] L. Subramanian, I. Stoica, H. Balakrishnan, and R. Katz, "Overqos: An overlay based architecture for enhancing internet qos," in *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation-Volume 1*. USENIX Association, 2004, pp. 6–21.

[26] E. Gilbert, "Capacity of a Burst-Noise Channel," *Bell System Technical Journal*, vol. 39, pp. 1253–1265, 1960.

[27] A. Konrad, B. Zhao, and A. Joseph, "Determining Model Accuracy of Network Traces," *Journal of Computer and System Sciences*, vol. 72, no. 7, pp. 1156–1171, 2006.

[28] X. Yu, J. W. Modestino, and X. Tian, "The Accuracy of Gilbert Models in Predicting Packet-Loss Statistics for a Single-Multiplexer Network Model," *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 4, pp. 2602–2612, March 2005.

[29] G. Hasslinger and O. Hohlfeld, "The Gilbert-Elliott Model for Packet Loss in Real Time Services in the Internet," *Proceedings of the 14th GI/ITG Conference on Measuring, Modelling and Evaluation of Computer and Communication Systems*, pp. 1–15, March-April 2008.

[30] M. Jelasity, S. Voulgaris, R. Guerraoui, A. Kermarrec, and M. Van Steen, "Gossip-based peer sampling," *ACM Transactions on Computer Systems*, vol. 25, no. 3, p. 8, 2007.

[31] L. Massoulié, E. Le Merrer, A. Kermarrec, and A. Ganesh, "Peer counting and sampling in overlay networks: random walk methods," in *Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*. ACM, 2006, pp. 123–132.

[32] L. Wilhelmsson and L. Milstein, "On the effect of imperfect interleaving for the Gilbert-Elliott channel," *IEEE Transaction on Communication*, 1999.

[33] C. Esposito, "Data Distribution Service (DDS) Limitations for Data Dissemination w.r.t. Large-scale Complex Critical Infrastructures (LCCI)," *Mobilab Technical Report (www.mobilab.unina.it)*, March 2011.

Christian Esposito is a research grant holder at the Institute for High Performance Computing and Networking (ICAR) - National Research Council of Italy (CNR). He graduated in Computer Engineering at Universitá di Napoli Federico II in 2006, and got his PhD at the same university in 2009. 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 regularly serves as a reviewer in several leading journals and conferences in the field of Distributed and Dependable Systems.

Marco Platania is a Postdoctoral Fellow at the Department of Computer Science - Johns Hopkins University (Baltimore, USA). He got the Ph.D. in Engineering in Computer Science in 2012 at Sapienza - University of Rome. His main research interests are P2P systems, publish/subscribe architecture, cloud computing and critical infrastructure protection. He regularly serves as a reviewer in several leading journals and conferences in the field of Distributed Systems.

Roberto Beraldi is an Assistant Professor at Dipartimento di Ingegneria Informatica Automatica e Gestionale, Sapienza - University of Rome, since 2002. He received the Laurea degree in Computer Science in 1991 and the Ph.D. degree in Computer Science in 1996 from the University of Calabria. He has published more than 60 peer-reviewed papers in various fields, including computer networks, wireless networks, and distributed systems. He also participates in many research projects and regularly serves as a reviewer for international conferences and journals in the above areas.