Improving the Efficiency of Gossiping

Christian Esposito
Institute for High Performance Computing and Networking (ICAR),
Napoli, 80131 - Italy
christian.esposito@na.icar.cnr.it

Roberto Beraldi
Dept. of Computer and Systems Engineering
University of Roma La Sapienza,
Roma, 00185 - Italy
beraldi@dis.uniroma.it

Marco Platania
Dept. of Computer Science
Johns Hopkins University,
Baltimore, MD 21218 - USA
platania@cs.jhu.edu

***Abstract*—In the current industrial practice, there is an increasing demand for an effective communication infrastructure to interconnect several heterogeneous systems. The effectiveness of the adopted communication infrastructure is defined in terms of the provided reliability and timeliness despite manifestations of failures within the network. Current approaches do not address both these aspects [7], but one aspect is generally assured at the expense of the other. Our driving idea is to take the best solution to achieve reliability and to improve its achievable performance so as to guarantee timely deliveries. Specifically, we propose to introduce determinism within the gossiping approach and to combine push and pull schemes. We have experimentally assessed such solutions through simulations, so as to find which is the one that best achieves both reliability and timeliness.**

***Index Terms*—Publish/Subscribe Middleware; Reliable Event Notification; Gossiping**

I. Introduction

Current software systems are characterized by a progressive increase in their scale and by a high demand for cooperation among their constituents, so as to adopt a "system of systems" perspective. The current literature is rich in practical examples of this novel generation of large-scale systems. The most demanding challenge that they have to face is to be able to interconnect several heterogeneous components in a large-scale setting, where the system is not limited within rack cabinets but spans over different distinct geographical sites and administrative domains. Therefore, the key component in these systems is the middleware solution adopted to enable the communication among their distributed components. Due to the required cooperation, the communication patterns that we can infer by analyzing the data flows within these systems do not consist of a naive request-respond communication style, where there is a client invoking a service upon a server and waiting for the result of such invocation. In most cases, we find a publish/subscribe communication style [1], where one, or even more, publisher asynchronously provides data to a set of interested subscribers.

The applications running on top of these systems present very strict non-functional requirements, *e.g.*, applications running on top of Grid or Cloud Computing typically can be assumed as business-critical, while the ones running on top of LCCI are mission-critical. Such application-level requirements are translated down to the middleware level in a set of proper constraints on the offered Quality-of-Service (QoS) in terms of reliability and timeliness. When considering large-scale systems, it is not practical to deploy a dedicated and proprietary network among interacting components. Therefore, communication can be only realized by means of the available IP-based network infrastructures, such as the Internet. However, such networks are typically affected by routing phenomena and failures that compromise the correctness of the packet delivery [2], which have a negative impact on the QoS experienced by users. Therefore, the adopted middleware has to be equipped with proper reliability enforcement methods to face such failures. In addition, the time to deliver information matters, since a message delivered too late can be useless or even dangerous for the system. Current approaches are not able to provide both reliability and timeliness, since recovering data dropped by the network typically implies some performance fluctuations. For a concrete example, a well-known approach called Gossiping [3] provides a high degree of reliability, while exhibiting a considerable worsening in performance. On the contrary, a distributed coding approach, called Network Coding [4], can present a more stable and predictable latency, while offering lower reliability guarantees.

Our driving idea is to select the best available solution to provide a high degree of reliability and to propose suitable methods to reduce its performance deficiency, so as to meet both reliability and timeliness. In a previous preliminary and theoretical work [5], we have shown that it is possible to combine the two mentioned approaches to obtain the best from both, *i.e.*, high reliability with no severe performance penalty. Such an intuition has been further proved by an experimental campaign in [6]. This previous work presents a significant flaw: the random nature of gossiping causes a high number of un-needed messages being exchanged among the nodes. This implies both a considerable traffic load on the network, which can cause congestion phenomena, and a non-optimal recovery time of lost data by wasting gossip messages sent towards nodes that do not require them. Our solution is to limit such random behaviour by forcing the protocol to prefer gossiping only with the nodes requiring a recovery action.

This paper is structured as follows. Section II provides a background on the current literature and describes our approach in combining coding and gossiping by highlighting open issues that we left untreated in our previous works. Section III presents our solution to introduce determinism, and Section IV proves the quality of our approach by means of simulations run in OMNET++. Last, we conclude with Section V, where we present the lessons learned with this work and its possible future evolution.

II. Background

As analyzed in details in [7], there are several approaches available in the current literature that can be used to provide reliability by tolerating data losses. On one side, we have approaches based on temporal redundancy, which use retransmissions to recover dropped messages. While, on the other side we have the one based on spatial redundancy, which send additional information along with application data. This applied redundancy is used to reconstruct the lost information without requiring any retransmission.

The best known example of a reactive approach for multicasting in large-scale systems is *Gossiping* [3]: a node stores a received message in a buffer with a size b, and forwards it for a limited number of times f_{in} (called fanin) to a randomly-selected set of nodes of size f_{out} (called fanout). Many variants of gossiping algorithms exist, which can be categorized as follows. In the *Push Approaches*, when a node receives a new message, it is immediately retransmitted to the randomly-selected nodes (*i.e.*, fanin is constant and implicitly equal to 1). On the other hand, in *Pull Approaches*, after a proper timeout t expires, nodes periodically send a list of recently-received messages. If a lost message is detected by comparing the received list with the history of messages received by the given node, then a retransmission of the lost message is requested. As we have experimentally shown in [6], these reactive solutions achieve a high degree of reliability. However, this gain is obtained at the cost of a reduced performance and timeliness, since latency exhibits severe fluctuations due to the high number of retransmissions needed to recover from consecutive losses, quite frequent in the current Internet [2].

A concrete example of a proactive approach is represented by *Network Coding* [4], which allows the generation of redundant information from the content of the application packets as a set of the linearly independent combinations. The benefit of proactive approaches is to reduce the performance worsening caused by the use of a reliability enforcement method. However, they offer a lower reliability degree since, if the redundancy degree is not properly set, data can be irremediably lost.

In [6], we have taken the gossip protocol, and enhanced it by introducing coding in two precise points of its algorithm: when data is transmitted disseminated among the members of a given group, and when data is gossiped to the randomly-selected nodes, *i.e.*, coding generates new packets both at the push delivery and at the pull-based retransmission. We have proved that a proper combination of gossip and coding is able to realize an optimal trade-off between reliability and timeliness with limited overhead worsening. Since the nodes receiving a gossip message are randomly chosen, there is a non-negligible probability that gossip messages may reach nodes that do not need them. To demonstrate this we have defined an utility function, namely U, and indicate a gossip message as useful if it is able to detect and recover a data loss. For a push gossip, U is the number of push messages that have allowed to recover a loss over the total number of received push messages per each node. For a pull gossip, U is the number of push calls that have triggered a retransmission over the total number of received pull calls per each node. In Fig. 1, we see that U has a low value, which further decreases when f_{out} is increased. This implies that a considerable part of the traffic generated by a gossip scheme is unneeded and optimizable, leaving space for improvements.

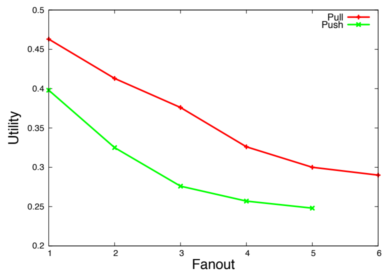


Fig. 1. Utility of the Gossip approaches from simulations in [6]

III. Determinism in Gossiping

We believe that the efficiency of gossip schemes in terms of performance and overhead can be improved by selecting nodes in a more deterministic manner. We refer to this approach as *Polarized Gossiping*, where node selection is not random, but based on a proper criterion. This solution is already present within the current literature, for example in [8] to reduce the overhead of gossiping in large-scale networks by allowing nodes to only gossip with other affine nodes, where node affinity is decided on the base of proximity or workload and information update frequency. However, our intention is different: we aim to speed up the recovery of lost data by preferring nodes with a high probability that the gossip message will be useful, so as to reduce the overall number of attempts to fully recover lost data.

We define as *optimal node selection* the approach to forward push messages only to those nodes for which such messages are useful. In the case of the push gossip, after the reception of a given event, a node will gossip with nodes still waiting for packets related to that event. Similarly, after the expiration of the timeout, a node will send a pull call message to the nodes that are still missing one, or even more, of the notifications contained in that pull call. Such a selection approach is not feasible in a large-scale system, since it requires complete knowledge of all the received notifications by each node within the system, which is not easily obtainable in most of the cases. Therefore, we propose proper heuristics for the node selection. Specifically, the driving idea is to assign a weight to each candidate to receive a gossip message, and those with the highest weights are selected for gossip. In particular, we describe two different methods for calculating such weights: in the first case, the weight is assigned based on the estimated overlay link status, expressed in terms of loss patterns; while in the second the weight assignment is based on the position of nodes within the tree. We conclude with several possible selection criteria based on the weights assigned to the nodes.

A. Labeling Nodes within the Multicast Tree

Our approaches for deterministic node selection require the knowledge of the topology within the multicast tree. This is typically a troublesome issue to be addressed in large-scale systems. However, we propose a method to handle it in a distributed and easy manner by assigning labels to each node. Specifically, each node has a label Φ of length l, where l is the level of that node in the tree. Each label is composed by l digits belonging to an m-ary alphabet, *i.e.*, a digit assumes a value in the range $[0; m-1]$. This label is assigned by the parent of the node: for each child in the tree, the parent adds a new progressive digit as the most significant one, while the root has $\Phi = null$. Labels are available to all the nodes thanks to a proper peer sampling service.

B. Weight Based on Loss Patterns

We assume that the links among nodes are not reliable, and exhibit a loss pattern characterized by the *Packet Loss Rate* (PLR), which is the probability of losing a packet. In particular, the adopted network model is the *Gilbert-Elliott* [9], one of the most-commonly applied in performance evaluation studies due to its analytical simplicity and the provided good results. PLR_x is continuously monitored by each node x over the overlay link that connects itself with its parent, *e.g.*, using the approach in [10], and disseminated towards the other nodes of the same group at the beginning, *e.g.*, after the node joins the group, and when there is a change in their value.

Such information on the link quality represents a valuable element for computing the weights for selecting nodes during a gossip round. Specifically, a given node x uses the PLR_x and the PLR estimated along the path between the root and its parent to compute the probability that it looses a packet along that path. We indicate this value with p_x, while $q_x = 1 - p_x$ is the probability of node x receiving a certain packet. Node x maintains information about all nodes along the path toward the root in a list called *ancestors* (easily computable in an iterative manner: when a node joins the tree, its parent passes to it its ancestor list). If the level of x in the tree is n, then the *ancestors* list contains n entries $z_0, z_1, ..., z_n$, where i indicates the level of a node in the tree and z_0 is the root and z_n is the node x itself. For each node z_i in this list, there is an associated packet loss rate PLR_{z_i} of the overlay link that connects z_i with its parent z_{i-1} (clearly, the packet loss rate associated to z_n is PLR_x, being $z_n = x$). Thus, the probability for node x of receiving a packet is the following one:

$$q_x = \prod_{i=1}^{n} (1 - PLR_{z_i}) \tag{1}$$

Such a value is continuously kept updated with the current value of PLR_{z_i}; if a node x communicates a change in its measured PLR, every node has to recompute its q_x, and relative weight, if the *ancestors* list contains the node x.

Based on these considerations, we define a formula that allows a node to assign weights to other nodes by considering the probability that those nodes have to lose packets. In the following, we describe how this formula is obtained for the previously introduced gossip strategies; then, we show how it can be easily modified for a given push/pull gossip strategy. The basic idea of this approach is that a node that receives a packet forwards it to the nodes that have a higher probability of having lost that packet. Thus, we denote with X the event *"node x received the packet"*, with Y the event *"node y received the packet"* and with $\overline{Y}$ the event *"node y did not receive the packet"*. Then, a node x that has to forward a received packet assigns to a node y a weight based on the following probability:

$$w_{xy} = Pr\{\overline{Y}|X\} \tag{2}$$

Such a value is influenced by the relative position of x and y in the tree. Let us consider x and y being on two completely different subtrees, *i.e.*, they share no common overlay links. Because we are assuming independent loss patterns, the probabilities of losing a packet at nodes x and y are totally uncorrelated. Thus, given two nodes x and y on two completely different branches of the tree, Equation 2 becomes:

$$Pr\{\overline{Y}|X\} = Pr\{\overline{Y}\} = p_y \tag{3}$$

If the node y is a predecessor of x, which is easily determined by comparing their labels, then obviously

$$Pr\{\overline{Y}|X\} = Pr\{\overline{Y}\} = 0 \tag{4}$$

Let us now consider nodes x and y in the same subtree. x has to find in the *ancestors* list the highest level ancestor h in common with y (it could be y itself if they are in the same branch of the tree. By indicating with n and l the level of y and h respectively, the probability of y having missed a packet knowing that x has received that packet depends on the probability of the packet having been lost during the path from h to y:

$$Pr\{\overline{Y}|X\} = \frac{Pr\{\overline{Y}, X\}}{Pr\{X\}} = 1 - \prod_{i=l+1}^{n} (1 - PLR_{z_i}) \tag{5}$$

It is easy to see that Equation 5 reduces to Equation 3 when the nodes x and y are on two completely different subtrees, the root of the tree being the highest level common ancestor (the root has $l = 0$). Indeed, in this case we have that $\prod_{i=l+1}^{n} PLR_i = p_x$. Thus, the final formula used by node x to assign a weight w_{xy} to node y includes the contributions 4 and 5:

$$w_{xy} = \begin{cases} 0 & \text{if y is predecessor of x} \\ 1 - \prod_{i=l+1}^{n} (1 - PLR_{z_i}) & \text{otherwise} \end{cases} \tag{6}$$

C. Weight Based on a Heuristic Approach

The previously described approach requires nodes to estimate the overlay link loss probability and to maintain additional information about the network conditions of their ancestors up to the root. In a large-scale system, it would generate a high network traffic, in addition to the scalability and consistency issues to maintain and update network status information. Thus, we also propose a heuristic approach to assign weights that imposes no additional burden on system nodes since it is only based on the topological information extracted from the labels of the nodes. Specifically, this information takes into account the level of a node in the tree, referred to as the *horizontal cut*, and the subtree it belongs to, referred to as the *vertical cut*. strategies.

The assignment of w_{xy} is based on two considerations. The first one is the Horizontal cut: nodes in a lower level are more useful because they are closer to the source. In fact, nodes at the bottom of the tree are expected to experience a higher number of lost packets. The second one is the Vertical cut: nodes in different subtrees are more useful because, with a high probability, they experience a different loss pattern. The higher is the level of the root of the subtree that contains the two nodes, the more useful is their interaction for a gossip procedure. These considerations are orthogonal and can combined to assign a weight to all the nodes. Let us consider two nodes x and y, with labels Φ_x and Φ_y and label sizes s_{Φ_x} and s_{Φ_y} respectively (remember that they represent their level in the tree). In addition, let us define $\rho_{(x,y)}$ the length of the common suffix of Φ_x and Φ_y. The formula used by node x to assign a weight to node y based on the *Horizontal cut* is

$$w^h_{xy} = 1 - \frac{s_{\Phi_x}}{s_{\Phi_x} + s_{\Phi_y}}, \tag{7}$$

while the assignment based on the *Vertical cut* is as follows:

$$w^v_{xy} = 1 - \frac{\rho_{(x,y)}}{s_{\Phi_x}}. \tag{8}$$

The rationales behind these formulas are the following ones. With respect to the Horizontal cut, Equation 7 contains a formula in the form of $f(x) = 1 - c/(c+d)$, with c a constant. Given the two nodes x and y, with x that assigns a weight to y, $c = s_{\Phi_x}$ and $d = s_{\Phi_y}$, this formula is such that the value of $f(x)$ reduces when the variable x decreases. $w_h(x, y)$ assumes a lower value when both s_{Φ_x} is high and s_{Φ_y} is low, *i.e.*, when node x is further from the root and, on the contrary, node y is closer to the source of the information. In this way, node x has a benefit when it contacts node y for a recovery attempt. With respect to the Vertical cut, Equation 8 represents the fraction of overlay paths not in common between the two nodes x and y. The higher is the value, the higher is the level of their common ancestor. When $w_v(x, y) = 1$, the common ancestor is the root, *i.e.*, the nodes are on two completely different branches of the tree.

The total weight that node x assigns to node y is simply the product of the two previous equations:

$$w_{xy} = w^h_{xy} \cdot w^v_{xy}. \tag{9}$$

D. Polarized Gossip

Thanks to the introduced concepts of weighted selection, we can formulate two new gossiping schemes. *Weighted Deterministic Polarized Gossip* (WDPG) selects the nodes among the ones with the highest weights. *Weighted Random Polarized Gossip* (WRPG) specifies that the nodes with the highest weights have the highest probability of being selected. We have noticed that all the nodes with the highest weights are placed at the bottom of the tree, so nodes at the higher levels may less frequently receive gossip messages. This can reduce the loss-tolerant capability of the polarized gossiping; therefore, we have designed two other schemes for a better distribution of gossip messages. In *Polarized Gossip with Window* (PGW), all the nodes follow the same criterion of WDPG, while certain nodes at the bottom of the tree (whose percentage is an algorithm parameter called window) do not select the nodes with the highest weights, but with the weights closest to the highest one. With PGW, we allow certain nodes to send gossip messages to nodes in other parts of the tree and not only at the bottom. *Pull+Push Polarized Gossip* (3PG) combines push and pull gossip schemes by using a push-based WDPG joint-ly with a pull-based WDPG, where during the pull rounds nodes are selected by considering the lowest weights and not the highest ones. Such an approach has also a window, which indicates the nodes that are not able to commence a push round. This is meant to reduce the overhead that characterizes the push gossip.

IV. Simulation Study

The scope of this section is to present experimental results that (i) study the impact of different node selection criteria on the quality of gossiping, and (ii) investigate the effect on the polarized gossiping when coding is used. To achieve this aim, we implemented our solution by using the OMNET++ (www.omnetpp.org) simulator, and decided not to use any real wide-area networks, such as PlanetLab, due to the uncontrollable loss patterns that make the obtained results non reproducible [11]. The workload has been taken from the requirements of the SESAR project, representative of a real critical large-scale system. Specifically, the exchanged messages have a size of 23 KB, the publication rate is one message per second and the total number of nodes is 40 (*i.e.*, the number of ATM entities involved in the first phase of the project). The network behaviour has 50 ms as link delay, and 0.02 as PLR, based on a measurement campaign described in [11], with message losses not independent as proved by [2]. We have assumed that the coding and decoding time are respectively equal to 5ms and 10ms. We have also considered the block size equal to 1472 bytes, so that an event is fragmented in 16 blocks. We have published 1000 events per each experiment, executed each experiment three times and reported the average.

The metrics evaluated in our study are the following. First, the *Success rate* is the ratio between the number of the received events and the number of the published ones, and this is referred to as the reliability of the publish/subscribe service. If the success rate is 1 (*i.e.*, complete reliability), then all the published events have been correctly received by all the subscribers. Second, the *Performance* is expressed as the mean latency, which is a measure of how fast the given dissemination algorithm is able to deliver notifications, and the standard deviation of the latency, which indicates the possible performance fluctuations due to the applied fault-tolerance mechanisms, highlighting the timing penalties that can compromise the

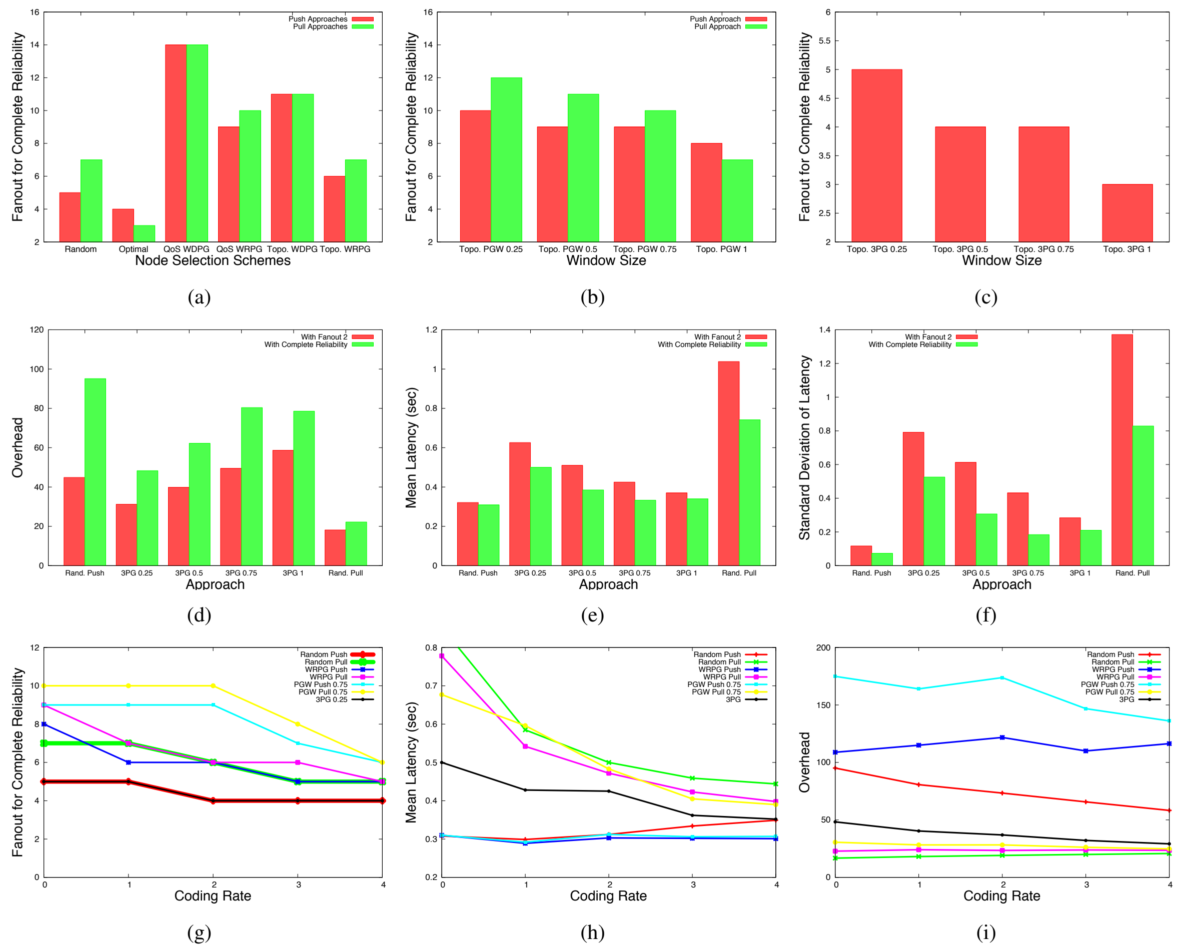


Fig. 2. Quality of the gossip scheme without and with coding.

timeliness requirement. Last, the *Overhead* is the ratio between the total number of datagrams exchanged during an experiment and the number of datagrams generated by the publisher. This is a measure of the traffic load that the dissemination strategy imposes on the network, and should be kept as low as possible, in order to avoid any congestion.

A. Polarized Gossiping

Let us compare the success rate achievable with random selection and the one with the introduced deterministic node selection schemes. A first consideration we can make is that the topology-based selection allows us to obtain a higher mean success rate than the QoS-based one. A second consideration is that polarized gossip, both WDPG and WRPG, allows us to have a lower mean success rate. The reason behind these two considerations is related to the fact that the topology-based heuristics and WDPG imply a strong focus on the nodes at the bottom of the tree, and losses in the higher levels of the tree are more troublesome to be recovered since they are selected less frequently. Such a focus on the nodes at the lower levels is less tight when applying a weighted random selection or if weights are computed with topology information, so that the gossip messages are more distributed, even if the nodes at the bottom still receive more gossip messages.

Let us consider the two schemes we have designed to deal with such issue. Fig. 2(b) shows that the windowing scheme in PGW is able to lower the required f_{out}; but it is not able to bring a considerable improvement than the cases with random selection. On the other hand, 3PG represents a better solution, thanks to its ability to forward pull calls towards the higher part of the multicast tree. From Fig. 2(c), we can notice that the f_{out} required by 3PG to obtain a complete reliability is lower than the one with a random selection. Moreover, lowering the window size implies a reduction of the obtainable success rate and a worsening of the convergence to a complete reliability.

Fig. 2(d) shows the experienced overhead for the gossip approaches with two bars: the first illustrates the overhead with a f_{out} equal to 2; while the second has the overhead when complete reliability is achieved. Generally speaking, there is a difference between these two bars, meaning that increasing f_{out} implies a growth of the experienced overhead. With the only exception being 3PG, the other node selection schemes do not present a lower overhead than the gossip with random selection, since their needed fanout is higher. As depicted in Fig. 2(d), lowering the window size allows us to have a lower overhead (even if the needed fanout is higher), considerably below that experienced with random selection, as shown in Fig. 2(d): at complete reliability, the overhead of 3PG with a window size of 0.25 is about half of the one with a random push (but is double than the one of random pull).

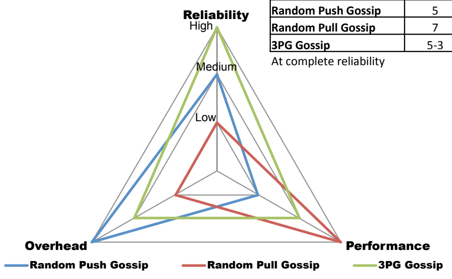
Let us consider the performance of the gossip schemes. Also, in this case we have two bars as in the previous one; however, in this case the second bar is always lower than the first, indicating that an increase of f_{out} brings a decrease in performance. Generally speaking, deterministic selection exhibits a performance below that achieved with random selection. However, such an improvement is nullified by the high number of f_{out} needed to have a complete reliability. Only 3PG is able to achieve a good performance. In particular, the increase of the window size in 3PG lowers the measured performance both in average and standard deviation (as depicted in Fig. 2(e) and Fig. 2(f)). So, with a window size of 1, we have that 3PG is very close to the performance of the random push gossip.

B. Polarized Gossiping with Coding

Coding can improve the quality of the polarized gossip approaches by exhibiting two kinds of benefits. First, when applied during dissemination it can help nodes closer to the root to recover lost events, which is not facilitated by the proposed deterministic selection modes. Second, when applied during gossiping it can speed up the event recovery by reducing the number of needed retransmissions. For these reasons, we have re-executed some of the experiments presented in the previous subsection by applying coding with a redundancy between 1 and 4 (we have considered only the topology-based weights, since coding has similar effects also on the QoS-based ones). For all the approaches, we have experienced a reduction of the value of f_{out} needed for complete reliability, as illustrated in Fig. 2(g). When random selection is applied, coding is able to reduce the needed f_{out} by about 25%. As expected, the improvement is more remarkable in the case of WRPG and PGW, since it is around 40%. 3PG has the same trend as random pull gossip. In Fig. 2(i), we can see a limited improvement of the overhead: the improvements for push gossip in terms of a lower f_{out} are nullified by the overhead introduced by coding. Fig. 2(h) shows an improvement of the experienced performance, even with the delay introduced by performing coding and decoding.

V. Conclusions and Future Work

The random selection of nodes in gossip schemes implies a low utility of the exchanged messages, causing some inefficiencies in terms of experienced latency and overhead. As depicted in Fig. 3, the random push has a higher loss-tolerant capability than random pull since it requires a lower f_{out} for complete reliability. This is achieved with a high overhead and a low performance costs, as opposite to random pull. We have proposed an improvement by presenting several schemes to realize what we called a polarized gossip. In particular, we have shown the pros and cons of such approaches. We have learnt that the best solution when determinism is introduced



	Fanout	Mean Latency	Overhead
Random Push Gossip	5	0,309	95,11
Random Pull Gossip	7	0,752	22,19
3PG Gossip	5-3	0,428-0,34	40,35-78,56

At complete reliability

Fig. 3. Schematic comparison.

in gossiping is to combine the push and pull approaches. As summarized in Fig. 3, such an approach has the highest loss-tolerant capability, and an optimal trade-off between overhead and performance. Deterministic approaches are further improved when coding is applied.

Acknowledgements

This work has been conducted when Christian Esposito was affiliated to the Department of Computer and Systems Engineering (DIS) at the University of Napoli Federico II, and when Marco Platania was affiliated to the Department of Computer and Systems Engineering (DIS) at the University of Roma La Sapienza. In addition, it has been partially supported by the Italian Project of National Research Interest (PRIN) DOTS-LCCI and by the BLEND Eurostar European Project.

References

[1] P.Th. Eugster et al. The many Faces of Publish/subscribe. *ACM Computing Surveys*, 35(2):114–131, June 2003.

[2] A. Markopoulou et al. Characterization of Failures in an Operational IP Backbone Network. *IEEE/ACM Transactions on Networking*, 16(4):749–762, August 2008.

[3] A.-M. Kermarrec, L-Massoulié, and A. J. Ganesh. Probabilistic Reliable Dissemination in Large-Scale Systems. *IEEE Transactions on Parallel and Distributed Systems*, 14(2):1–11, February 2003.

[4] R. Ahlswede, S.-Y.R. Ning Cai Li, and R.W. Yeung. Network Information Flow. *IEEE Transactions on Information Theory (TIT)*, 46(4):298–313, July 2000.

[5] 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 1st International Workshop on Network Resilience*, October 2011.

[6] C. Esposito, S. Russo, R. Beraldi, M. Platania, and R. Baldoni. Achieving Reliable and Timely Event Dissemination over WAN. *Proceedings of the 13th ICDCN, - Lecture Notes in Computer Science*, 7129:265–280, January 2012.

[7] C. Esposito, D. Cotroneo, and S. Russo. On reliability in publish/subscribe services. *Computer Networks*, 57(5):1318–1343, 2013.

[8] I. Gupta, A.-M. Kermarrec, and A. J. Ganesh. Efficient and Adaptive Epidemic-Style Protocols for Reliable and Scalable Multicast. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 17(7):593–605, July 2006.

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

[10] C. Fragouli and A. Markopoulou. A network coding approach to overlay network monitoring. *Allerton*, 2005.

[11] C. Esposito. Data Distribution Service (DDS) Limitations for Data Dissemination w.r.t. Large-scale Complex Critical Infrastructures (LCCI). *Mobilab Technical Report*, 2011.