DOI: https://doi.org/10.1016/j.future.2018.05.026

On the optimal tuning and placement of FEC codecs within multicasting trees for resilient publish/subscribe services in edge-IoT architectures

Christian Esposito *, Andrea Bruno, Giuseppe Cattaneo, Francesco Palmieri

Department of Computer Science, University of Salerno, Via Giovanni Paolo II, 132, I-84084, Fisciano (SA), Italy

A B S T R A C T

Publish/subscribe services represent the key choice to glue together the huge amount of heterogeneous devices available within the Internet of Things, by making them interoperable at a large scale through mediation systems and facilities available in the cloud. In such a scenario, several optimizations have been introduced in the architectural layout by pushing parts of the service intelligence away from centralized points to the logical extremes, namely the edge, of the infrastructure, according to the edge computing paradigm. Such services can establish tree-based overlays among the involved nodes, and are used on top of unreliable networks, where packets have a non-negligible probability to be lost. In order to reduce such losses affecting wide area communications, the need to move data towards the cloud has been reduced by placing more frequent computations at the edge of the infrastructure, nearer to data sources, according to the edge computing paradigm. However, this is not enough and, in order to provide resilient and reliable communications, publish/subscribe implementations have been equipped with means to achieve loss-tolerance, which unfortunately have been proved to be ineffective (by lacking having end-to-end guarantees) and inefficient (by compromising the communication performance). In this paper, we identify Forward Error Correction as a suitable method to have efficient and effective loss tolerance within multicast trees, and describe how dealing with its issues by having the interior nodes within the multicast trees to generate spatial redundancy in addition to the one produced by the root. The decision of which nodes on the network edge must generate the additional redundancy and how many additional packets must be forwarded has been approached by using a Single-Leader Multi-Follower Game. Such an approach has been empirically assessed and compared with a centralized one, represented by a genetic algorithm, and with gossiping, so as to show the achievement of optimal decisions.

Keywords:
Publish/subscribe
Game theory
Forward error correction
Sensor communications
Single-leader multi-follower game

1. Introduction

The Internet is progressively transforming into a huge-scale complex ubiquitous and pervasive computing ecosystem, integrating billions of fixed and mobile networked objects deployed over wide geographic areas, that continuously produce and consume significant amounts of data, and are mutually interconnected according to several communication paradigms and technologies grouped under the Internet of Things (IoT) umbrella. In this scenario, the flexibility of cloud-based architectures, together with the recent advancements in the area of distributed computing and data storage give the opportunity of shifting the more computationally expensive and storage-demanding operations to the cloud in order to benefit of scale economies, as well as increased flexibility and elasticity in the resource management. However, such a

* Corresponding author.
E-mail addresses: esposito@unisa.it (C. Esposito), andbruno@unisa.it (A. Bruno), cattaneo@unisa.it (G. Cattaneo), fpalmieri@unisa.it (F. Palmieri).

shift introduces several challenges affecting traditional network-centric architectures, ranging from the astonishing demand for processing and storage resources, often exceeding the capability of traditional cloud solutions, to the impossibility of collecting the large volumes of data generated by millions of sensing devices in a central location for timely processing and analysis or conveying continuous flows of control information back to these devices. The edge-computing paradigm has been developed to face these challenges, by pushing intelligence, processing power and communication capabilities within specialized devices located on local area networks serving IoT nodes, thus as closer as possible to the data producers/consumers. Such devices can also perform message brokering and assume a fundamental role in reducing the potentially large volumes of control traffic returning back from centralized applications and systems located within the cloud, by using optimized communication techniques mainly for multicasting within the context of IoT-specific information exchange service paradigms, such as the publish/subscribe one.

Publish/subscribe services [1] are middleware solutions implementing flexible message passing, where events associated to a node assuming the role of publisher are notified to other observing nodes, known as subscribers, according to their specific interests (explicit subscriptions) by means of decoupled and asynchronous interactions. These event notifications are mediated by brokers, which are special nodes properly routing incoming notifications in order to pass them towards the interested subscribers, *i.e.*, those that have previously expressed some subscriptions satisfied by these notifications. Brokers can run on special edge nodes with enough computing and storage resources, in order to perform some kind of in-network processing actions on notifications [2], that may also assume the role or publishers and/or subscribers, depending on the underlying architecture and data exchange patterns. Several kinds of communication protocols and transport facilities can be adopted by publish/subscribe services to realize the distribution of notifications from publishers to the interested subscribers through the brokers. Specifically, such services can be realized by exploiting the communication capabilities of traditional unicast transport-level protocols, such as TCP; but they can also use transport- or network-level multicast and/or broadcast facilities as the ones offered by UDP or raw IP. When scalability is a must, the typical strategy is to establish a proper overlay among publishers, subscribers and brokers [3], which forms a multicast tree, so as to have a decentralized architecture for distributing notifications in an optimized way. Due to the high degree of scalability and flexibility provided by their intrinsic decoupling and decentralization features, these services have been extensively used within many industrial projects aiming at realizing large-scale IoT infrastructures, spanning across different application domains, ranging from health information systems or smart cities, to air traffic control or monitoring railway signaling systems.

Several use cases where publish/subscribe services have faced increasing interest within the IoT arena are characterized by not only a strong demand for scalability and flexibility guarantees, but also by other non functional properties [4], among which reliability is of pivotal importance. With respect to the reliability, such services are strictly required to always deliver published notifications to all the interested subscribers, or to none of them, so as to provide a consistent and coherent event delivery service tolerating the possible faults that can occur within the nodes or along the network. The last aspect related to networking failures and anomalies is particularly serious for large-scale systems due to the use of the Internet, where packet loss events may have a non negligible loss probability and length (in terms of the number of consecutively lost packets) depending on the network conditions [5]. In fact, when working with the Internet, even if the involved traffic is assigned to the most privileged QoS class, we have to assume that the underlying network infrastructure is inherently dynamic and unreliable, due to the lack of a unique administration/management authority that ensures the availability of communication paths, connectivity resources and devices. Moreover, publish/subscribe services are mainly based on connectionless transport protocols, such as UDP, directly relying on the bare packet delivery mechanisms provided by IP at the network-layer. Despite the undeniable gains in terms of performance and scalability, such a choice has to deal with a "best effort" delivery that is not able to cope with network error conditions. For this aim, most of the available solutions realizing reliable event notification within the IoT have been equipped with a proper application-level scheme to recover from packet losses due to the above network problems [6]. All the widely applied schemes are based on acknowledgments and retransmissions [7], ensuring a high degree of reliability in the whole delivery process.

Despite being able to provide a high degree of success rate in delivering notifications, independently of the loss patterns exhibited by the network, such a solution is not efficient in terms of experienced performance, and things get worse when the scale grows. Some use cases of publish/subscribe service do not only impose a guaranteed delivery under any possible network conditions, but also consider keeping message delivery within proper time constrains as a demanding endeavor. For a concrete example, a flight controller receiving an outdated notification with the position of an aircraft can lead to a wrong decision and the false positive detection of a collision. For this reason, it is important to realize the needed trade-off between reliability and timeliness within event notification, and to adopt solutions able to achieve it also in IoT domains characterized by a huge number of devices. Forward Error Correction (FEC) [8] is potentially a good candidate for such purpose due to its independence from the network conditions and the possibility of having a bounded worsening of the communication performance. However, such a solution is typically not scalable, since applied in a centralized manner with a single node generating the FEC redundant data used to recover the lost packets without triggering a retransmissions [9]. In [10], a decentralized approach has been proposed as a trade-off between the traditional centralized deployment, and a distributed one represented by network coding, where all the brokers generate additional redundancy. Such a solution is the perfect trade-off between the fine-grained control over the FEC redundancy (provided by a distributed approach) and the bounded costs (offered by a centralized approach). However, to concretely adopt such a solution it is crucial to deal with the problem of deciding which broker has to be a codec within a specific multicast tree, in order to optimize the usage of edge nodes, and how many FEC packets are needed to guarantee the delivery. In [11,12], such a problem has been approached by means of typical game theory techniques, which demonstrated to be quite effective. However, the unordered strategy decision taken by each node, may cause a considerable transitory before reaching the stable solution represented by the Nash Equilibrium. In this work, we adopted a different approach by leveraging the sequential nature of the problem, so as to use a Single-Leader Multi-Follower Game. The obtained solution will be compared with the one obtained with a centralized resolution method based on a genetic algorithm so as to prove the reach of optimal decisions that can reveal to be crucial in large scale edge-IoT architectures.

The remaining of the paper is structured as follows. Section 2 provides the needed background on resilient publish/subscribe services, and introduces the problem of providing timely and resilient notification delivery by tolerating the loss patterns exhibited by the network with the use of the spatial redundancy produced by some of the interior nodes within a tree. Section 3 describes the solution to the introduced problem by using a centralized and distributed approaches, while Section 4 shows results

obtained from the experimental assessment of the introduced solutions. The paper ends with Section 5 presenting the lesson learnt and a plan for future work.

2. Background and related work

A publish/subscribe service is made of several types of applications: publishers that produce notifications about the occurred events (*i.e.*, a change in their internal state or received information from environmental monitoring), and subscribers that consume notifications of their interest. The work of joining publishers and subscribers is done by a set of brokers [6], which can be run at the same nodes where the publishing and/or subscribing applications have been deployed, or even being hosted on special machines, such as edge nodes within a modern IoT infrastructure. We can abstract such a case by assuming that the infrastructure is composed of a set N of nodes, which are interconnected by a proper set of directed links from the set L, where $l_{i,j}$ indicates the link from the i-th node to the j-th one. Each interior node, *i.e.*, a node with an incoming link (e.g. the uplink coming from the uppermost part of the hierarchy) and a certain number of outgoing links (e.g. reaching its served devices), is characterized by hosting a broker that receives notifications from the incoming link, duplicates and forwards them along the outgoing links to the other nodes. A leaf node, *i.e.*, a node with no outgoing links, and hence a terminal IoT device, such as a controllable sensor or actuator, is the one that does not present any broker, but models a subscriber. In the overlay, nodes represent subscribers and brokers, but not publishers, since any publishing operation consists in sending notifications to a specific node so that they travel throughout the overlay and reach all the nodes [13]. The links among the nodes are established based on the mutual interest of the subscribers; in this work, we are interested in topic-based publish/subscribe schemes, which are simple to implement, exhibit stable and good performance and have found a wide acceptance in the current practice. In particular, all the subscribers interested to the same topic establish an overlay thanks to the mediation role of the brokers.

The node overlay can assume two kinds of topologies. On the one hand, there are tree-based approaches [14], which are characterized by a structured organization of the nodes realizing a tree, where each node can implicitly define its parent, from which it receives the incoming messages, and children, to which it dispatches the incoming messages. On the other hand, there are mesh-based approaches [15], which expose a less structured organization letting each node to employ a swarming delivery mechanism to a certain subset of nodes. Mesh-based approaches are more resilient to process crashes and network disconnections since they do not have a rigid structure and can more easily adapt to changing conditions. However, they are more complex to be built and maintained than trees since nodes need to have knowledge of some of the other ones, which is challenging to obtain within Internet-scale systems. For this reason, in this work we have focused on a tree-based publish/subscribe service, whose concrete example is Scribe [13], a large-scale and decentralized publish/subscribe solution based on the Plaxton-based DHT capabilities of Pastry [16] to construct and maintain a multicast tree among the subscribers interested to the same topic.

The Internet is characterized by a non-negligible probability to lose a packet, namely Packet Loss Rate (PLR), and when a packet is lost it is possible to also lose some of the subsequent ones, whose average proportion is named Average Burst Length (ABL). We intend with loss pattern the pair of PLR and ABL characterizing the lost packet process along a link $l_{i,j}$, namely $\lambda_{i,j} = \{PLR_{i,j}, ABL_{i,j}\}$. The link quality degradation and the consequent intensification of the loss patterns are mainly caused by physical problems on the networking infrastructure and by the consequent BGP updates [17,18]. In addition, the augmented loss rates can be due to bulk message delivery [19], inducing network-wide congestion, and routing malfunctioning or communication carrier errors [20]. A resilient publish/subscribe service is able to tolerate the loss patterns within the multicasting tree and to deliver the published notifications to all the subscribers within the tree [6]. The naive solution of using TCP in order to have a resilient message delivery is not acceptable, since, despite of the scalability and flexibility issues [6], such a solution is only able to offer link-by-link guarantees, *i.e.*, the message forwarded by an end-point of a single link of the overlay is guaranteed to reach the other end-point, but not end-to-end guarantees across the whole tree, *i.e.*, a published notification is guaranteed to reach all the designed subscribers by traversing the overlay multicast tree. Therefore, a set of proper application-level solutions have been designed and realized on top of connectionless transport protocols, such as UDP. The rich literature of these solutions can be divided in two main groups based on the type of redundancy introduced: the ones using temporal redundancy or retransmissions with a reactive behavior, *i.e.*, detecting a loss and recovering it with a consequent retransmission, and the ones using spatial redundancy where the behavior is more proactive by sending additional data so as to reconstruct what missed by a packet loss without triggering any retransmission. Within the first class, we find the widely-known Automatic Repeat-reQuest (ARQ) scheme [7], which is used by TCP for reliable unicast communications, and the gossiping [21], where there is no single node responsible for storing messages and managing the retransmissions, as the tree root or the parent node in the ARQ, but such an operation is distributed among all the nodes of the overlay. Despite the reactiveness of these solutions allow them to achieve a high degree of resiliency, they also exhibits a degraded throughput and increased latency, as shown in [9]. There have been attempts to improve them in order to reduce their costs and drawbacks [22,23], especially within the context of sensor networks. However, when timeliness matters, the solutions using spatial redundancy are a better option, however, they lack of offering strong resiliency guarantees: if not properly tuned, they are not able to guarantee the successful delivery of notifications to all the interested subscribers within the tree since the applied redundancy is not able to reconstruct what has been lost. Examples of the aforementioned solutions are Forward Error Correction (FEC) [8], where additional packets are obtained by encoding the notifications to be exchanged and are used to reconstruct the lost packets, as well as path redundancy [24], where multiple paths are established, so that the notifications to be exchanged can take more than one route to reach a subscriber. Path redundancy is very effective in circumventing node and link failures, but it is not similarly effective in the case of packet losses, *i.e.*, losses on different paths cannot be uncorrelated with each other [25] since it is hard to have multiple paths that do not share any networking device. In addition, path redundancy is more successful in wireless sensor networks rather than in the Internet-based hybrid communication infrastructures, due to the difficulty of guaranteeing path diversity [26]. Internet-based large scale IoT architectures are characterized by plenty of redundant paths in the core, but few paths to be exploited at the network edge, even in the case of multi-homed LANs, so that diversity is not fulfilled. For this reason, in our specific reference scenario, we have investigated the use of FEC-based solutions.

FEC-based solutions can be implemented according to centralized and distributed schemes, with one or more nodes generating the additional packets, namely *codecs*, and the redundancy tuned according to the demands of the subscribers. In a centralized scheme, the only codec is the root of the multicast tree, namely r, which applies a redundancy equal to the worst loss pattern experienced when reaching one of the destinations:

$$\rho_r \geq \max_{i=0,\ldots,S} \lambda_{r,i}, \tag{1}$$

where S is the total number of subscribers within the tree, ρ_r is the redundancy introduced by r and $\lambda_{r,i}$ is the loss pattern between r and the ith destination. Such a solution is simple to realize, but ineffective, due to the poor scalability and control of the applied redundancy. Specifically, when the loss patterns of the links along the multicast tree are not balanced, with a portion of the tree having heavy losses and the other portion experiencing limited losses, the generated redundancy by the central codec must be tailored on the heavy losses, causing a large number of unneeded packets to flow throughout the portion with limited losses. This problem is named as "the boat unbalanced by the heaviest" (BUH), and requires a more fine-grained control of the introduced redundancy. More distributed schemes, as the Network Coding [27], consist in all the interior nodes, *i.e.*, those nodes in the tree to have at least one child node, being codecs and generating a redundancy equal to the worst loss pattern along the links towards its children:

$$\rho_i \geq \max_{j \in C_i \subset N} \lambda_{i,j} - \rho_{in}, \tag{2}$$

where C_i is the set of nodes directly reachable by the i-th node with its outgoing links, and ρ_{in} is the redundancy that the i-th node has received from the previous node, *i.e.*, its parent. While being able to deal with the BUH problem, this solution is not efficient since it implies a high performance worsening. A different solution is the one named as Network Embedded FEC [10], where only a subset of interior nodes are selected as codecs, with a redundancy generated by considering the worst loss pattern along the paths towards the other codecs or subscribers in the lower levels of the multicast tree, and the received redundancy from the previous codec, namely ρ_{in}:

$$\rho_i \geq \max_{j \in U_i} \lambda_{i,j} - \rho_{in}, \tag{3}$$

where U_i is the set of the underlying codec in the tree with respect to the i-th node. This solution allows a fine-grained control of the applied redundancy and do not incur in the BUH problem, at the cost of a moderate performance burden. However, it is not a simple task to properly decide which interior nodes should be selected as codecs. This issue has been formalized as an optimization problem known as the *P-Median Problem*, with the exception being the number of median which has not been fixed a priori, where the objective is to find the number of needed codecs and their locations by minimizing the applied unneeded redundancy and the costs to set up a codec:

$$min \sum_{i \in U} \sum_{j \in L} d_{i,j} x_{i,j} + \sum_{j \in L} \alpha_j, \tag{4}$$

subject to:

$$\sum_{j \in L} x_{i,j} = 1, \quad \forall i, \tag{5}$$

$$x_{i,j} \leq y_j, \quad \forall i, j, \tag{6}$$

$$\sum_{j \in L} y_j = p \leq n = |L|, \tag{7}$$

$$x_{i,j}, y_j \in \{0, 1\}. \tag{8}$$

where the transportation cost $d_{i,j}$ is the redundancy that the codec in location j has to generate according to Eq. (3), and α_j represents the costs of placing a codec in the jth location. This is the problem for which we study a scalable solution in large scale edge-IoT environments.

3. A distributed approach to codec placement for scalable IoT scenarios

The solution of the codec placement and tuning problem can be centralized at a single node of the tree, or distributed among all the nodes. Different existing solutions are described in the remaining of this section and a novel game theory-based fully distributed approach is proposed in order to achieve the needed scalability and applicability in the reference IoT scenario.

3.1. Defining an upper bound: A centralized greedy solution implemented with a genetic algorithm

Within the multicast tree, one node, *e.g.*, the root, is responsible of planning the placement and tuning the codecs by collecting the demands of the subscribers, resolving the optimization problem in Eq. (4), and instructing the other nodes based on the obtained solution. Such an operation can be done once, at the beginning, or continuously based on the dynamics of the loss patterns experienced by the links of the tree. Moreover, this P-Median optimization problem is known to be *NP-Hard* on general networks for an arbitrary p (where p is a variable) and polynomial when the network is a tree [28]. Therefore, heuristics are often used in practice when dealing with large problem instances. We have examples of this approach within the current literature, such as in [10], where a greedy algorithm is used by the decision node. Such an algorithm starts with an empty solution, and adds a codec at a time, until all the codecs have been placed within the multicast tree. However, recent approaches for solving a P-Median problem tend to use meta-heuristics, since with the classic methods of local searches the solution quality may deteriorate rapidly together with the problem size [29].

Clearly, a centralized solution is able to achieve better performance (*i.e.*, returning a solutions lying on the Pareto front of the set of the admissible solutions, or very close to it) than any kind of distributed scheme, due to its specific nature, essentially characterized by a immediate and total availability of network status information. However such a kind of solution is not realistically deployable in a large scale scenario, since a single node is not able to collect a complete and clear view of the network information within a large-scale system, the scale of the optimization problem can be extremely overwhelming, and the time for converging to an optimal solution can be large. Accordingly, in order to obtain a baseline/upper bound to be used as a reference for evaluating the performance of distributed solutions, we implemented the greedy approach proposed in [10] by using a genetic algorithm. Genetic algorithms [30] represent a class of metaheuristics widely adopted in the literature of Operational Research, since they are fast search mechanisms due to the possibility of considering more than one solution at a given iteration. Genetic algorithms are based on the concept of evolution, and they are structured in three cyclic steps, as shown in Fig. 1(b): *evaluation*, *ranking*, and *evolution*. Specifically, the algorithm builds an initial set of solutions Z, called population; then, it evaluates their quality (defined as *fitness*). Solutions are ranked based on their assigned fitness value, and the best ones are selected and saved within an archive. If a termination condition is not reached, a new population is obtained by applying evolutionary operators to the current population. Specifically, a chromosome is the representation of a solution to a given optimization problem solved by a genetic algorithm. The structure of a chromosome for the codec localization problem is depicted in Fig. 1(a) and composed of a binary part and a permutation one. The binary part indicates the number of codecs to be placed within the multicast tree, namely ν; while, the second part contains all the possible locations where a codec can be placed. In particular, the ν codecs in the given solution are collocated in the first ν locations

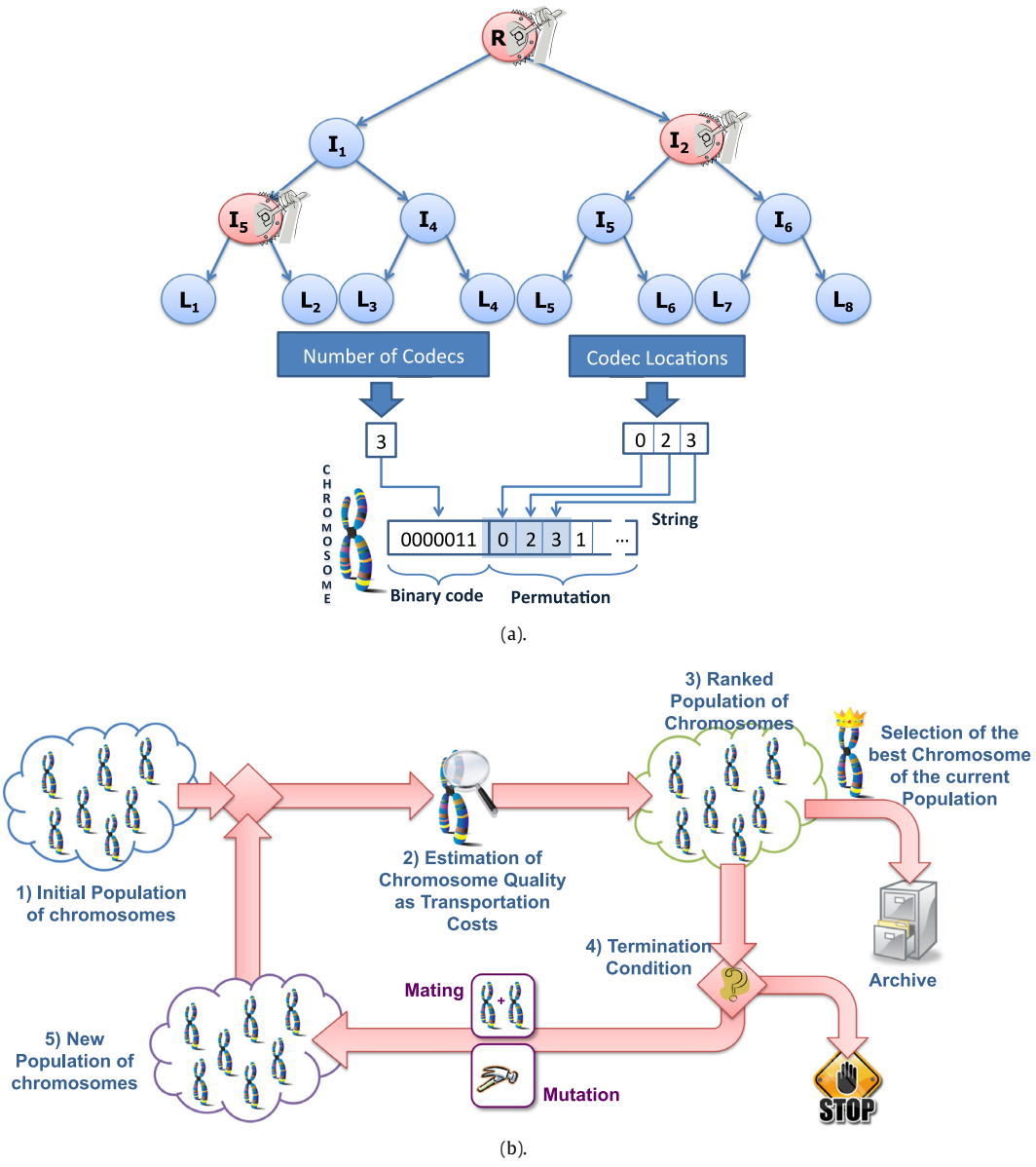

Fig. 1. Representation of a chromosome for solving the codec finding problem and schematic overview for a genetic algorithm.

contained in the permutation part of the chromosome. Having two different parts in the chromosome means that we need to have two different definition for the typical evolutionary operators: one for the binary part, and one for the permutation one:

- *Mating* — given two starting solutions, they can be combined to generate two novel solutions:
  - *One-point cross-over* for the binary part: A single cross-over point, namely π, is selected. All binary digits beyond that point in both chromosomes are swapped between the two parent chromosomes;
  - *OX cross-over* for the permutation part: Two cross-over points, namely π_1 and π_2, are randomly selected, and everything between the two points is swapped.
- *Mutation* — a solution is altered so as to obtain a new one:
  - *Bit-flip* for the binary part: a point of mutation, namely μ, is randomly selected, and the binary digit in that place is changed to the opposite value, according to a given probability;
  - *Reciprocal exchange* for the permutation part: two points of mutation, namely μ_1 and μ_2, are randomly selected. The order of everything between the two points is inverted, according to a given probability.

The typical termination condition is a maximum number of iterations, or the impossibility of finding a new solution that outperforms the ones already stored in the archive within a given number of iterations. In our work, we have assumed the second one as the termination condition by setting up a threshold on the number of iterations when an update of the archive has not been done from the best solutions from a given population. Last, an update of the solution archive does not only consist in storing the best solutions of a population, but also to check if the novel solution does not have a dominance relationship with the old ones. Specifically, a given chromosome dominates another one if it holds an higher fitness value. Then, when updating the archive, the new solutions are inserted, afterwards all the solutions within the archive are ordered based on their fitness value and the ones with a lower value than the highest one are removed. This allows to keep in

the archive (and to return to the user after the termination of the approach) only the best solutions found during the algorithm iterations.

Although a centralized solution allows simplifying the problem resolution and taking optimal decisions, as above mentioned, it is unfeasible in a large-scale distributed system, as the one addressed by this work, since collecting global information is extremely difficult, if not impossible in such a context. In addition, even if assuming possible to acquire any global knowledge about the infrastructure, it still exhibits serious scalability limits that prejudice its usage for systems composed by a large number of nodes. In fact, the time to collect loss statistics in a centralized point linearly increases with the number of nodes within the multicast tree. Moreover, the memory required to store all the collected data and/or the load to resolve the optimization may overwhelm the resources of the central decision node, making impossible to find a solution.

3.2. A distributed solution with a single-leader multi-follower game

A distributed approach has the strength to overcome the applicability issues of the previous solution for large-scale systems, since codec placement is performed by using local computations at each node of the multicast tree; hence avoiding turning to a central decision maker with global knowledge of the system. However, this advantage is paid at the expenses of obtaining a placement that is slightly far away from the optimal one (*i.e.*, the found placement is characterized by a non-negligible distance from the Pareto front). The authors of the above mentioned centralized solutions with a greedy algorithm have extended their previous work in a distributed manner, as presented in [31], by using local decisions and negative acknowledgments, namely *NACKs*, to elect new codecs and/or to adjust the redundancy degree applied by the chosen codecs. When disseminating a message between two nodes, the destination has to forward a *NACK* message to notify the source that the message has not been correctly delivered. Specifically, a message cannot be delivered if the number of lost packets is greater than the correcting capacity of the adopted FEC techniques or the destination has received packets belonging to a message with an identifier greater than the expected one. Such a message is also used to indicate the number of lost packets. Each node in the tree is characterized by a variable named *UNREACHED_KIDS*, which indicates the number of its children that could not receive the published message, and another variable called *LOSS_LENGTH*, which contains the maximum number of the consecutively-lost packets communicated by each of its children. When a node receives a *NACK* message, it increases the current value of *UNREACHED_KIDS* and update *LOSS_LENGTH* (as specified in rows 4–5 in the algorithm). When an interior node, able to decode the received message, is not acting as a codec and has the *UNREACHED_KIDS* not null, it becomes a candidate to be a codec. Such candidate is chosen as a codec if its *UNREACHED_KIDS* is greater than a given threshold σ, otherwise it notifies its codec with its value of *LOSS_LENGTH* and asks to increase the applied redundancy. In case the node has been elected as codec, it starts applying to the incoming messages an additional redundancy degree equal to *LOSS_LENGTH*, and it can even ask the previous codec to reduce its redundancy. A node also monitors if the redundancy degree applied by its codec is a waste, in this case it can issue an *UPDATE* message with a negative value in the field length, so that when the codec makes the update, it would result in a decrease of the applied redundancy.

The decision of a codec placement with the previously-described distributed method is made only considering the number of the reached nodes and not on the generated redundancy. To this aim, the codec placement is not optimized by considering Eq. (4). As a practical example, let us consider the multicast tree in Fig. 2, where the losses experienced along each edge is indicated. Consider the application of the solution from [31], with σ equal to 1 and opening costs of a codec equal to the maximum loss pattern, *i.e.*, 7. At the first iteration, no redundancy is applied and all the interior nodes miss the message. At the second iteration, the tree root becomes a codec (since the number of unreached children is greater than 1), and applies a redundancy equal to 3. However, with such a situation, some children of the interior nodes still miss the message. At the third iteration, all the interior nodes becomes codecs, since all of them have more than one unreachable child. Therefore, the algorithm reaches the final placement, where both the root and the interior nodes acts as codec, with a cost of 35 (computed by summing the redundancy received by all the nodes). Let us consider a different placement, where the root applies more redundancy than the one needed by its direct children, *e.g.*, 4, and having only the interior nodes 1 and 3 acting as codec. In this case, the total cost is minor, and equal to 31. Such a simple example showed us the inefficiency of the previous algorithm in finding the best solution for our codec placement problem.

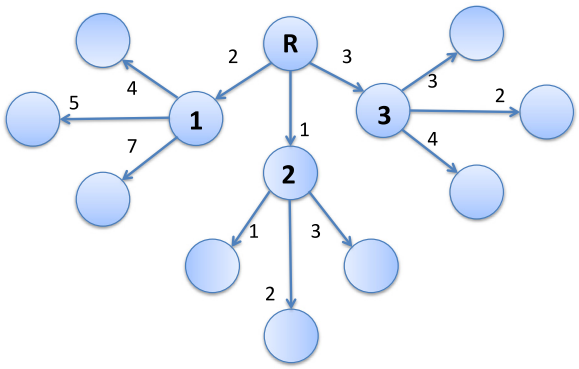

Fig. 2. Example multicast tree.

To deal with this issue, a better solution is to map the problem of placing codecs onto a multi-agent system [32]: the multicast tree is populated by agents that receive inputs from the environment and react according to a proper strategy and certain local information so as to collectively solve the problem. The cooperation and self-organization capabilities of the agents allow the resolution of the codec placement problem and result suitable to approach distributed optimization problems without requiring any global control [33]. During the last decades, several multi-agent systems have been proposed and some of them have been applied for the resolution of location problems, which have been an active research topic within the context of game theory [34], leading to the so-called facility location games. Within a multi-agent system with game theory inspired heuristics, the agents pursue well-defined exogenous objectives by considering their knowledge and expectations of the behavior of the other agents. In [11,12], the non-cooperative formulation has been applied as follows. There are n players (where n is the number of interior nodes within the multicast tree), whose strategy is represented by a binary decision to play the role of codec or not. If the node has to act as a codec, Eq. (3) is applied to determine the redundancy to be generated. In order to drive the players towards a solution, the global cost function specified in Eq. (4) is decomposed in a set of local cost functions assigned to each player [35]:

$$C_i(S_i) = \alpha_i \cdot S_i + (\rho_i + \max_{j \in U_i} \lambda_j) \cdot (1 - S_i), \tag{9}$$

where ρ_i is the redundancy received by the i-th node, $\lambda_{i,j}$ is the loss pattern experienced by the j-th node having the i-th node as parent

Table 1
Codec Placements for the example in Fig. 2.

Placement	Codec id	Applied redundancy
Solution from [31]	Root	3
	Interior node 1	6
	Interior node 2	1
	Interior node 3	4
Improved solution	Root	4
	Interior node 1	5
	Interior node 3	3

within the multicast tree, and α_i is the cost to pay if the player has chosen to act as a codec. Specifically, the game starts with players picking up their strategy at random from their admissible strategy set, and evolves over the time where each player changes its strategy so as to minimize its costs. Such an evolution continues until none of the players is willing to change its strategy, so that the game reaches a stable codec placement represented by the Nash Equilibrium (NE). The existence of NE solutions has been shown in [11] by referring the work in [36] and indicating two conditions to be guaranteed:

$$\exists i \in Y \text{ s.t. } \rho_i + \max_{j \in U_i} \lambda_j \leq \alpha_i \tag{10}$$

$$\nexists i \in Y \text{ s.t. } \rho_i < \alpha_i - \max_{j \in U_i} \lambda_j \tag{11}$$

Despite having tried to better formalize this solution by using distributed strategic learning in [37,38], it is known that a non-cooperative formulation is not optimal, due to the lack of coordination among the players within the game (see Table 1). Another source of error in such a formulation is that all the players potentially take their decisions in the same instance, however, in our problem we can notice a leader–follower scenario, where the node at the higher level within the tree, or leader, should have all the knowledge about the reaction of the underlying nodes and take its decision before the nodes in the lower level, which are called the followers. The followers see the decision of the leader and based on it determine which strategy to implement. According to such a vision, the overall multicast tree can be decomposed into a series of overlapping local groups formed by a leader and multiple followers, as shown in Fig. 3. In addition, the nodes should not take their decisions only once, but the overall decision process must be repeated over multiple time periods by considering the changing demands. The possible resolution method is derived from the classic Stackelberg game [39], which involves a single leader ad a single follower for just one period. In our case we have multiple followers, that play a Nash game with each other as indicated within the work presented in [11,12], while the leader plays a Stackelberg game with the followers based on their expected reactions, as indicated in [40]. More specifically, the cost of the leader to be minimized is equal to the cost underlying its decision (*i.e.*, associated to following the selected strategy), added to the cost given by the best response functions of the follower according to the two conditions in the Eqs. (10) and (11) (*i.e.*, the redundancy applied by the i-th node added to the maximum demands of its children is greater than α) and indicated with $f_{b\_r}(j, S_i)$ for the j-th node when its parent decides for the strategy S_i. Formally, we can model the decision process of the leader as the minimization of the following augmented cost function:

$$C_i^s(S_i) = C_i(S_i) + \max_{j \in U_i} C_i(f_{b\_r}(j, S_i)), \tag{12}$$

The leader will perform a local search within the set of its admissible strategies by looking for the one with the minimum cost according to the previous Equation. Such a formulation has the ambition of achieving better solutions than the ones achievable

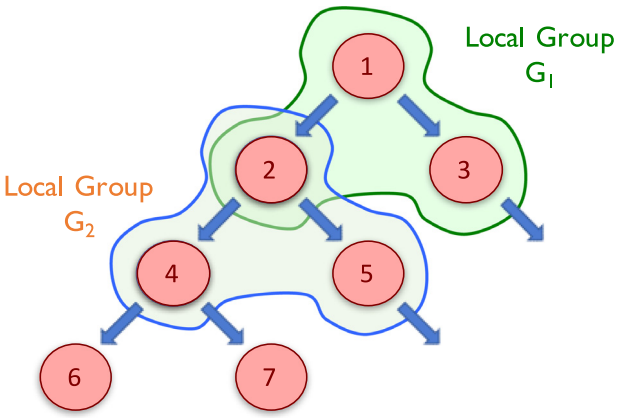

Fig. 3. Segmentation of the multicast tree.

by the non-cooperative formulation in [11,12], since it introduces a sort of collaboration among the nodes since the leader of each local group predicts the response of the followers by considering their possible selected strategies. Such a methodology is more effective than a cooperative re-formulation of the solution in [12] by means of a Nash Bargaining Solution (NBS) [41], where initially all the players follow the non-cooperative game, and only after a certain timeout (when the NE is eventually reacher) an iterative bargaining is started. In fact, right from the beginning a sort of indirect cooperation among the nodes is promoted, so that the convergence to a cooperative solution is obtained before the NBS.

4. Experimental evaluation

The main goal of this section is to present some experimental results to compare the effectiveness of the proposed distributed algorithm against the baseline centralized scheme proposed in [10] and described in Section 3.1. Specifically, we have implemented an ad hoc Java program that builds up a multicast tree, and simulates the delivery behavior and the loss patterns along the links among the nodes. We have explicitly decided to avoid implementing a real prototype to be run on a real testbed like PlanetLab [42,43], in order to have a more controllable testing platform and achieve easily reproducible results. The root of the tree has to pass along the multicast tree messages with a size of 23 KB, properly fragmented in packets with a size of an Ethernet MTU (1472 bytes), so that a single event is fragmented in exactly 16 packets. The links among the nodes are characterized by a 50 ms link delay, a 0.02 PLR, and a 2 ABL (all the $\lambda_{i,j}$ values are set equal to $(0.02, 2)$. The coding and decoding times have been imposed equal to 5 ms and 10 ms, respectively. Finally, without loss of generality, we considered a system with 1 publisher and 39 subscribers, all subscribed to the same topic, *i.e.*, members of the same multicast tree. We have published 1000 events for each experiment, executed each experiment 3 times and reported the average results.

In the first set of experiments with 40 nodes within the multicast tree, we have compared the best effort guarantees provided by (i) the tree with a bare solution providing no resiliency, (ii) the codec planning scheme proposed in [10] implemented with a centralized GA running at the root as presented in Section 3.1, (iii) the non-cooperative game formulation proposed in [11,12], and, finally, (iv) the novel formulation presented in this work with the single-leader multi-follower Stackelberg game.

During the experiments, three metrics of merit have been considered: (i) Resiliency, obtained as the mean ratio of successfully received notifications over the total number of published notifications; (ii) Latency, determined as the mean time needed by

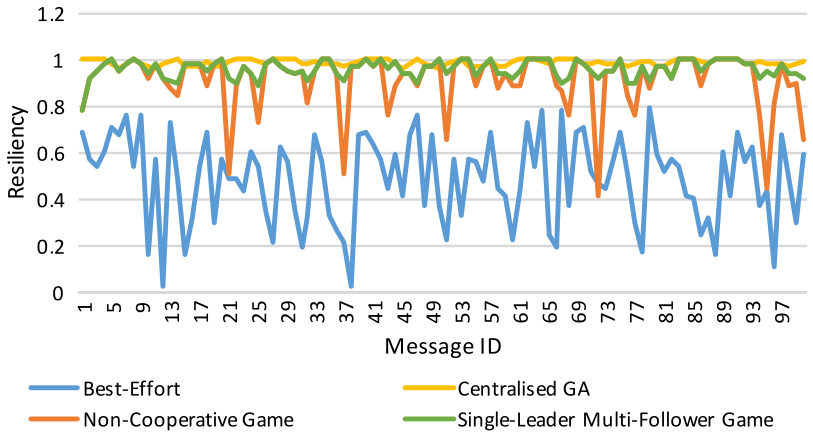

Fig. 4. Resiliency comparison over time (messages sent) and 40 nodes.

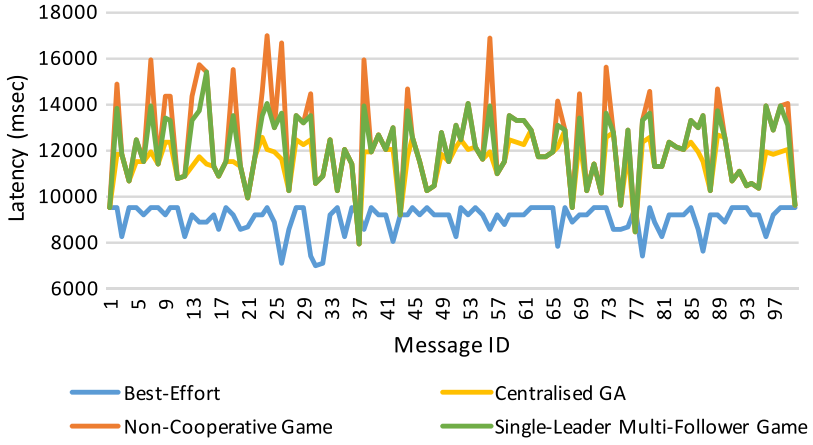

Fig. 5. Latency comparison over time (messages sent) and 40 nodes.

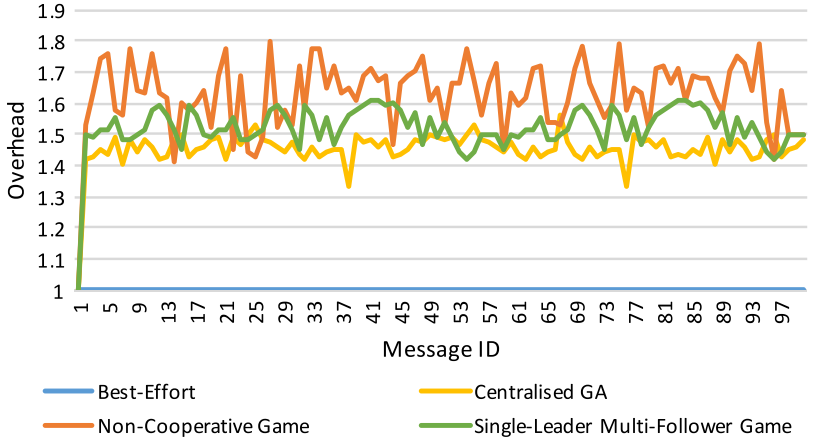

Fig. 6. Overhead comparison over time (messages sent) and 40 nodes.

the notifications to move along the tree from the root to each interested subscriber; (iii) Overhead, computed as the mean ratio of exchanged packets along the links over the total number of 16 packets needed to convey each message with a size of 23 KB.

Fig. 4 shows the resilience achieved by the 4 approaches, and in contrast with the severe losses experienced by the best-effort delivery mode, all the other solutions are able to increase their resiliency, with the optimal placement obtained by the centralized GA being the best one since its result is very close to the value of 1, indicating that all the interested subscribers have received all the published notifications. Presumably, the lack of the complete resiliency is caused by the high variability of the loss patterns so that the optimal tuning based on the measured losses may be not valid in the following time period since losses may have been increased slightly. However, it should be considered, that a centralized scheme is not practically applicable in a large scale edge-IoT scenario, due to the motivations presented in the previous sections. If we consider the other two approaches, our single-leader multi-follower Stackelberg game-based approach is able to obtain a better value of resiliency, since the obtained placement

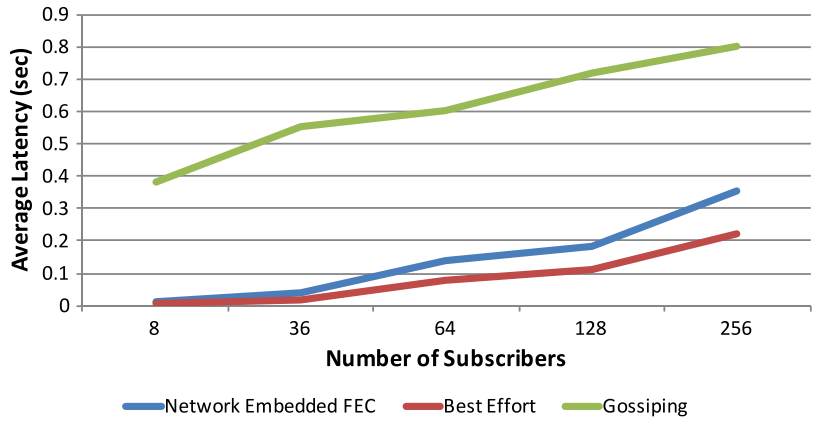

Fig. 7. Latency comparison with varying number of nodes.

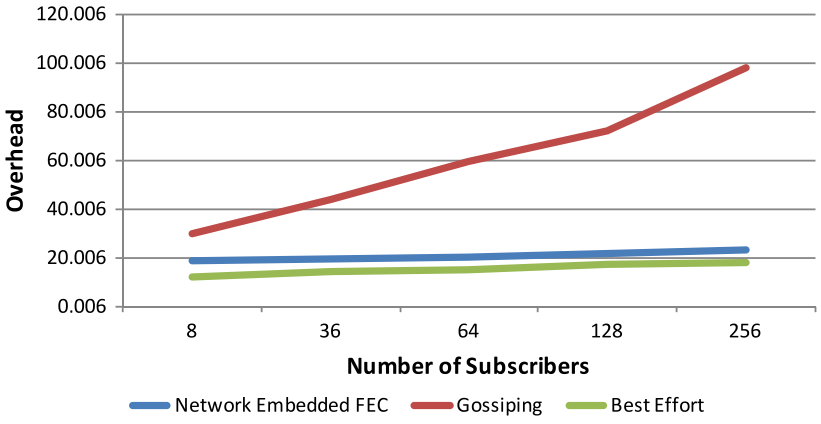

Fig. 8. Overhead comparison with varying number of nodes.

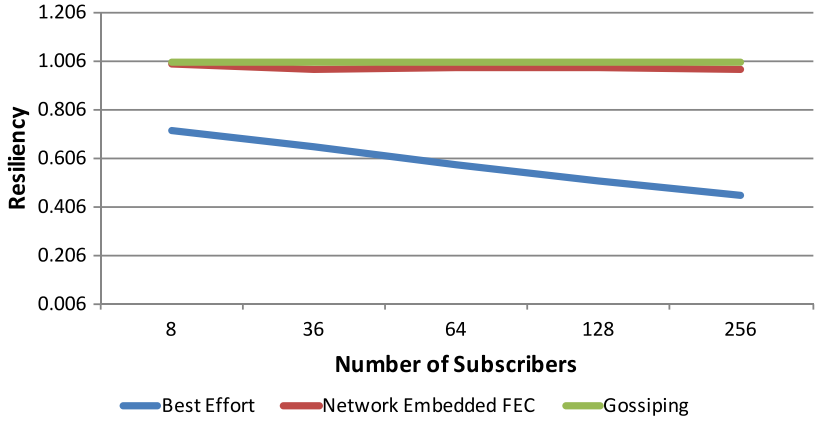

Fig. 9. Resiliency comparison with varying number of nodes.

is of higher quality respect to the one of the non-cooperative game formulation. Such a difference is also noticeable in Fig. 5 and Fig. 6, where the better performing solutions are able to lower respectively the mean latency and the overhead.

In the second set of experiments characterized by an increasing number of nodes within the multicast tree, we have compared the best effort delivery strategy with our single-leader multi-follower Stackelberg game for codec placement and the gossiping. From Fig. 7, we can notice that gossiping with pull-based mode, (*i.e.*, when a node receives a notification, it randomly selects some nodes, different than its children, to forward a list of the received notifications in order to detect losses and trigger retransmissions) is the one implying a strong performance worsening due to the delay of detecting losses and requesting retransmissions. Our approach has a latency closer to the reference one, due to the use of best-effort delivery. From Fig. 8, gossiping with push-based mode (*i.e.*, when a node receives a notification it randomly select some nodes, different than its children, to forward the received notification), exhibits a severe overhead, able to cause congestion phenomena. On the contrary, our solution is more effective by implying a more bounded overhead, slightly influenced by the scale of the service. Last, Fig. 9 indicates the resiliency, where

gossiping shows the highest degree possible, despite the particular mode being used. Our approach is close to achieving such a high resiliency degree, but some packets can be still lost, due to the loss patterns variability and unpredictability.

5. Concluding remarks

This paper has presented the problem of providing resilient event notification within the context of a tree-structured topic-based publish/subscribe without incurring excessive performance worsening. The proposed approach has been to adopt a decentralized FEC scheme, where a subset of the interior nodes within the multicast tree performs coding operations to generate FEC spatial redundancy. Such an approach requires the optimal placement and tuning of the codecs, which consists in a P-median optimization problem to be centrally approached with a genetic algorithm and in a distributed manner by using game theory. We have showed the effectiveness of our proposed solutions by means of preliminary experiments by simulating a multicasting tree and comparing the illustrated solutions. For a future work, we aim at continuing the investigation of the single-leader multi-follower Stackelberg game by introducing the uncertainty in the announced experienced loss patterns, due to the volatility of the network behaviors and the possibility of having unstable loss patterns over the time.

References

[1] P.Th. Eugster, P.A. Felber, R. Guerraoui, A.-M. Kermarrec, The many faces of publish/subscribe, ACM Comput. Surv. 35 (2) (2003) 114–131.

[2] C. Esposito, A. Castiglione, F. Palmieri, M. Ficco, C. Dobre, F. Pop, Event-based sensor data exchange and fusion in the internet of things environments, J. Parallel Distrib. Comput. 118 (part 2) (2018) 328–343.

[3] M. Hosseini, D. Tanvir, S. Shimohammadi, N.D. Georganas, A survey of application-layer multicast protocols, IEEE Commun. Surv. Tutorials 9 (3) (2007) 58–74, Third Quarter.

[4] P. Bellavista, A. Corradi, A. Reale, Quality of service in wide scale publish/subscribe systems, IEEE Commun. Surv. Tutorials 16 (3) (2014) 1591–1616, Third Quarter.

[5] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, Y. Ganjali, C. Diot, Characterization of failures in an operational IP backbone network, IEEE/ACM Trans. Netw. 16 (4) (2008) 749–762.

[6] C. Esposito, D. Cotroneo, S. Russo, On reliability in publish/subscribe services, Comput. Netw. 57 (5) (2013) 1318–1343.

[7] S. Lin, D. Costello, M. Miller, Automatic-repeat-request error-control schemes, IEEE Commun. Mag. 22 (12) (1984) 5–17.

[8] L. Rizzo, L. Vicisano, RMDP: an FEC-based reliable multicast protocol for wireless environments, ACM SIGMOBILE Mobile Comput. Commun. Rev. 2 (2) (1998) 23–31.

[9] C. Esposito, M. Platania, R. Beraldi, Reliable and timely event notification for publish/subscribe services over the internet, IEEE/ACM Trans. Netw. 22 (1) (2014) 230–243.

[10] M. Wu, S.S. Karande, H. Radha, Network-embedded FEC for optimum throughput of multicast packet video, J. Signal Process.: Image Commun. 20 (8) (2005) 728–742.

[11] C. Esposito, A. Castiglione, F. Palmieri, M. Ficco, A game-theoretic approach to network embedded FEC over large-scale networks, in: Proceedings of the International Symposium on Intelligence Computation and Applications, ISICA 2015, in: Communications in Computer and Information Science book series (CCIS), vol. 575, January 2016, pp. 353–364.

[12] C. Esposito, A. Castiglione, F. Palmieri, M. Ficco, Building a network embedded FEC protocol by using game theory, Inf. Sci. 433–434 (April) (2018) 365–380.

[13] M. Castro, P. Drushel, A.M. Kermarec, A. Rowstrom, Scribe: A large-scale and decentralized application-level multicast infrastructure, IEEE J. Sel. Areas Commun. 20 (8) (2004) 1489–1499.

[14] Y. Chu, S.G. Rao, S. Seshan, H. Zhang, A case for end system multicast, IEEE J. Sel. Areas Commun. 20 (8) (2002) 1456–1471.

[15] N. Magharei, R. Rejaie, Y. Guo, Mesh or multiple-tree: A comparative study of live P2P streaming approaches, in: Proceedings of the 26th IEEE International Conference on Computer Communications, INFOCOM 07, May 2007, pp. 1424–1432.

[16] A. Rowstrom, P. Drushel, Pastry: Scalable, decentralized object localization and routing for large-scale peer-to-peer systems, in: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms, in: Lecture Notes in Computer Science, vol. 2218, November 2001, pp. 329–351.

[17] F. Palmieri, U. Fiore, A. Castiglione, F.-Y. Leu, A.D. Santis, Analyzing the internet stability in presence of disasters, in: Security Engineering and Intelligence Informatics: Proceedings of the ARES 2013 Workshops: MoCrySEn and SeCIHD, 2013, pp. 253–268.

[18] F. Palmieri, Percolation-based routing in the internet, J. Syst. Softw. 85 (11) (2012) 2559–2573.

[19] Y. Ran, Considerations and suggestions on improvement of communication network disaster countermeasures after the wenchuan earthquake, IEEE Commun. Mag. 49 (1) (2011) 44–47.

[20] D. Reina, S. Toral, F. Barrero, N. Bessis, E. Asimakopoulou, Modelling and assessing ad hoc networks in disaster scenarios, J. Ambient Intell. Humanized Comput. 4 (5) (2013) 571–579.

[21] P.T. Eugster, R. Guerraoui, A.-M. Kermarrec, L. Massoulié, Epidemic information dissemination in distributed systems, IEEE Comput. 37 (5) (2004) 60–67.

[22] C. Esposito, A. Castiglione, F. Palmieri, M. Ficco, Improving the gossiping effectiveness with distributed strategic learning (invited paper), Future Gener. Comput. Syst. 71 (June) (2017) 221–233.

[23] A. Bhatia, R.C. Hansdah, TRM-MAC: A TDMA-based reliable multicast MAC protocol for WSNs with flexibility to trade-off between latency and reliability, Comput. Netw. 104 (July) (2016) 79–93.

[24] S. Birrer, F. Bustamante, A comparison of resilient overlay multicast approaches, IEEE J. Sel. Areas Commun. 25 (9) (2007) 1695–1705.

[25] D.G. Andersen, A.C. Snoeren, H. Balakrishnan, Best-path vs. multi-path overlay routing, in: Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement, October 2003, pp. 91–100.

[26] J. Han, D. Watson, F. Jahanian, An experimental study of internet path diversity, IEEE Trans. Dependable Secure Comput. 3 (4) (2006) 273–288.

[27] T. Ho, M. Medard, R. Koetter, D.R. Karger, M. Effros, J. Shi, B. Leong, A random Linear Network Coding Approach to Multicast, IEEE Trans. Inform. Theory 52 (10) (2006) 4413–4430.

[28] B.T. abd R.L. Francis, T. Lowe, Location on networks: a survey. Part I: The P-center and P-median problems, Management Sci. 29 (4) (1983) 482–497.

[29] N. Mladenovic, J. Brimberg, P. Hansen, J.A. Moreno-Perez, The p-median problem: A survey of metaheuristic approaches, European J. Oper. Res. 179 (3) (2007) 927–939.

[30] R. Haupt, S. Haupt, Practical Genetic Algorithms, Wiley, July 2004.

[31] M. Wu, H. Radha, Distributed codec placement algorithm for network-embedded FEC, in: Proceedings of the 40th Annual Conference on Information Sciences and Systems, pp. 151–156, 2006.

[32] M. Wooldridge, N. Jennings, D. Kinny, A methodology for agent-oriented analysis and design, in: Proceedings of the Third Annual Conference on Autonomous Agents, AGENTS, 1999, pp. 69–76.

[33] S. Moujahed, O. Simonin, A. Koukam, Location problems optimization by a self-organizing multiagent approach, Multiagent Grid Syst. 5 (1) (2009) 59–74.

[34] M. Osborne, A. Rubinstein, A Course in Game Theory, The MIT Press, 1994.

[35] N. Li, J. Marden, Designing games for distributed optimization, IEEE J. Sel. Top. Sign. Proces. 7 (2) (2013) 230–242.

[36] C. Chekuri, J. Chuzhoy, L. Lewin-Eytan, J. Naor, A. Orda, Non-cooperative multicast and facility location games, IEEE J. Sel. Areas Commun. 25 (6) (2007) 1193–1206.

[37] C. Esposito, A. Castiglione, F. Palmieri, F. Pop, Distributed strategic learning for building network embedded coding to achieve reliable event notification, in: Proceedings of the 2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech), August 2016, pp. 360–367.

[38] C. Esposito, A. Castiglione, F. Palmieri, F. Pop, Distributed strategic learning and game theoretic formulation of network embedded coding, J. Comput. Sci. (2018) (in press). Available online at http://www.sciencedirect.com/science/article/pii/S1877750317303058.

[39] H.D. Sherali, A.L. Soyster, F.H. Murphy, Stackelberg-nash-cournot equilibria: characterizations and computations, Oper. Res. 31 (2) (1983) 253–276.

[40] T. Başar, R. Srikant, A stackelberg network game with a large number of followers, J. Optim. Theory Appl. 115 (3) (2002) 479–490.

[41] M. Goemans, M. Skutella, Cooperative facility location games, J. Algorithms 50 (2) (2004) 194–214.

[42] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, M. Bowman, PlanetLab: an overlay testbed for broad-coverage services, ACM SIGCOMM Comput. Commun. Rev. 33 (3) (2003) 3–12.
[43] N. Spring, L. Peterson, A. Bavier, V. Pai, Using PlanetLab for network research: myths, realities, and best practices, Oper. Syst. Rev. 40 (1) (2006) 17–24.