Gossip-Based Monitoring Protocol for 6G Networks

Mauro Femminella, *Member, IEEE*, and Gianluca Reali

***Abstract*—The service function (SF) area has gained increasing attention in the last years due its ability to combine the advantages of cloud computing with network softwarization. By decoupling SFs from the physical equipment where they are executed, it is possible to make network services scalable and flexible. These advantages become even more evident in the forthcoming 6G networks, where the overall environment is expected to become more dynamic and cloud-based, with SFs deployed as cloud-native functions. However, in order to efficiently manage and compose services using these SFs, it is necessary to monitor the available resources of the nodes where they can be deployed, in addition to exchange information relevant to the operational status of active SFs. To this aim, we propose a lightweight monitoring architecture by using agents in charge of monitoring the status of SFs running in co-located clusters. These monitoring agents exchange their information by means of a gossip protocol, which allows increasing the reliability of the process. In this way, it is possible to keep service decisions as local as possible, limiting the interactions with centralized decision and orchestration platforms, and thus increasing network scalability and responsiveness. Performance evaluation shows the effectiveness of the proposed solution, and demonstrates that the network overhead of the distributed monitoring process is definitely affordable.**

***Index Terms*—6G, network signaling, network discovery, network monitoring, gossip protocol.**

I. Introduction and Background

THE DEVELOPMENT of the 6G systems requires facing significant technological challenges in different directions [1]. For example, wireless link throughput needs to be scaled up to realize breakthrough applications such as holographic communication based on interactions with so-called digital twins [2]. A distributed computing system is necessary to manage and process a growing volume of data exchanged through massive connectivity that characterizes the so-called Internet of Everything (IoE). For this reason, not only an intense and ubiquitous use of the edge computing model [3] is envisioned for the deployment of a myriad of new applications, but the overall 6G architecture is expected to evolve towards a wide-area cloud, encompassing the wireless, edge, and core segment, as well as data centers [4]. However, this complexity requires that the network management and control planes have to be specifically designed to address a massively distributed architecture, thus requiring to:

- Make heavy use of artificial intelligence (AI) and machine learning (ML) computational models to learn how to configure, optimize and heal themselves, instead of relying on pre-planning procedures only [5];
- Be intrinsically secure, and implement advanced embedded trust models such as the Blockchain [6];
- Intensively leverage cloud computing services in order to make the whole 6G system cloud-native and optimized for ubiquitous computing [4].

Therefore, control and management planes face scalability problems with a complexity that scales up by orders of magnitude compared to 5G. This complexity can hardly be managed by approaches that concentrate the vision of the network and the management of information in a centralized way. Instead, the control and decision layer are expected to be distributed all over the 6G wide-area cloud [4]. The three technological pillars identified above (distributed intelligence, distributed trust model, and cloud-native system infrastructure) share the need for a data distribution protocol [7] able to convey data in a timely and resource efficient way in highly distributed systems. In this paper, we propose and analyze a solution addressing this requirement based on one of the main models of distributed information sharing, namely *gossiping* [8], which is known for its intrinsic scalability and robustness. A solution based on gossip protocols can be used:

- To transport information to enable decentralized learning in highly distributed AI-based environments [9], [10];
- As consensus protocol for Blockchain applications [11];
- To share information related to the service functions (SFs), which run in the 6G network to compose advanced services. This *monitoring function* allows building a fully decentralized orchestration of network services.

In this paper, we focus on a gossip-based solution to implement information sharing of monitoring information related to the SFs. Nevertheless, it can be adapted without major modifications also to provide the other two functions. In more detail, we refer to a system adopting network softwarization technologies, providing on-demand networking and computing resources by decoupling SFs from physical nodes where they run. In 6G, these technologies are pushed forward with respect to 5G, thanks to its highly distributed and cloud-native architecture [4], moving from network softwarization (SDN and NFV [12]) towards intelligence softwarization [13]. The introduction of serverless technologies, especially in edge nodes [14], has increased the dynamic nature of the instantiated SFs. In fact, the continuous instantiation and removal of

Manuscript received 15 July 2022; revised 15 December 2022 and 15 February 2023; accepted 26 March 2023. Date of publication 31 March 2023; date of current version 12 December 2023. This work has been carried out in the framework of the projects APE5G, funded by University of Perugia, and 5G-CARMEN, funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825012. The views expressed are those of the authors and do not necessarily represent the project. The Commission is not responsible for any use that may be made of the information it contains. The associate editor coordinating the review of this article and approving it for publication was K. Xue. *(Corresponding author: Mauro Femminella.)*

The authors are with the Department of Engineering, University of Perugia, 06125 Perugia, Italy, and also with the Consorzio Nazionale Interuniversitario per le Telecomunicazioni, 43124 Parma, Italy (e-mail: mauro.femminella@unipg.it).

Digital Object Identifier 10.1109/TNSM.2023.3263542

microservices though serverless deployment, orchestrated by increasingly sophisticated AI-based tools, makes it challenging to bring knowledge of the local state of service deployment to a higher level. This problem, mitigated in 5G systems with hierarchical orchestration systems [15], is exacerbated in 6G [16]. It is due to the increasing number of nodes where these (micro)SFs can be deployed [17], [18] and their heterogeneity, being computing resources available not only in data center clusters, but also in edge and even user equipment nodes, creating the so-called edge-to-cloud continuum [19], [20], [21]. Therefore, a continuous communication exchange with a central orchestrator (CO) would limit responsiveness to variations of service load, determining the generation of significant volumes of management traffic and the consequent increase in network overhead.

Addressed problem: According to the distributed approach to resource management expected in 6G [4], [5], computing resources of the wide-area 6G cloud are organized in clusters, each with its own local orchestrator (LO). An LO is in charge to take informed resource management decisions, keeping the decision process as local as possible. A CO is deployed to have a centralized point of interaction with LOs and a more abstracted view of underlying resources, as well as a collector for specific information, such as models' parameters in distributed AI approaches. A fundamental component for such a distributed architecture is the monitoring and data distribution function, which provides an up-to-date view of the status of resources in the network. Classic approaches based on publish/subscribe platforms with a central broker, such as Kafka, do not work well, as communication has to go back and forth from the CO [22], [23].

Contributions: The contribution of this paper is twofold. First, we introduce an architectural solution supporting a distributed monitoring process between computing clusters. It is based on the use of a local monitoring agent (MA), co-located with each orchestrator. The MA is charge to retrieve the status and service capability of SFs and to exchange these data with other peer MAs. In this way, each LO can see an updated picture of the whole network or slice [3]. Second, we propose a gossip protocol to implement the distribution system among MAs to provide a distributed and robust solution, which can easily address the dynamic nature of 6G networks. To setup the gossip overlay between the MAs, they make use of a discovery function *embedded* in the monitoring protocol itself, without requiring a further mechanism to accomplish this task. The proposed gossip protocol leverages packet interception capabilities, enabled by network softwarization technologies, to improve operations efficiency.

This paper significantly extends our preliminary conference paper [24], presenting a more complete architectural view as well as a thorough performance evaluation. This includes a performance model for the proposed solution and the comparison with other up to date proposals, including a centralized pub/sub one taken from the recent literature [22], [23], [25], not present at all in the preliminary version.

The paper is organized as follows. In Section II, we analyze the related work in the field. In Section III, we present our gossip protocol for distributing monitoring data. The performance analysis is illustrated in Section IV. Finally, in Section V we draw our conclusions.

II. Related Works

A formalization of the gossip problem is proposed in [8]. As most of gossip solutions, our proposal is round-based. Gossip rounds can be synchronous or asynchronous. Synchronous ones need a synchronization system that increases overhead. For this reason we propose an asynchronous approach. Gossip protocols generally select peers involved in a gossip session randomly, although also gossip protocols using deterministic strategies exist [26]. Gossip protocols can either involve a single pair of peers, or multiple separated pairs, or multiple overlapping pairs. In this regard, an important feature of our proposal is the capability to establish gossip sessions with multiple peers in a single round, which allows saving bandwidth and provides multiple system updates within a single gossip round, thus lowering the time between information updates. To a certain extent, this approach can be compared with the gossip algorithms used in wireless multi-hop/ad-hoc networks. In fact, the gossip protocol in [27] leverages the broadcast nature of the wireless medium to send messages to all neighbors in a single round, whilst our solution exploits the packet interception capabilities of SDN devices for delivering gossip messages to multiple peers in a single round.

Gossip-based solutions can be used to solve the discovery problem [28], [29]. Some analogies between our proposal and the one shown in [28] exist, although the problem formalization is different. Both algorithms aim to create a network spanning tree, used for distributing messages. Nevertheless, the proposal in [28] needs of a prior knowledge of all node interfaces in the network for creating a spanning tree. Even if the tree generation is started by an arbitrary node, this tree is used for distributing messages over the entire network. On the contrary, our proposal is fully distributed, and each peer runs the same algorithm and creates its own distribution tree. As for network discovery, the *the two-hop walk* in [29] is quite similar to the D-mode discovery process proposed in paper [30], using randomized gossip as well. However, it assumes prior knowledge of the set of neighbors. A solution for collecting this information is proposed in [31].

It is worth to mention also the off-path signaling protocol (OSP) [32], specifically designed for providing a signaling framework for NFV. However, the OSP proposal still retains the two-layer organization of NSIS protocol suite [33], adding gossip-based peer discovery and peer-to-peer flooding message distribution. We show that this organization is not strictly necessary, and that it is possible to embed the monitoring function in an enhanced gossip function, with significant saving of bandwidth, as well as protocol simplification.

There has been a renewed interest in gossip protocols in recent years, driven by the distributed nature of 6G architectures and involved technologies. Most of proposals regarding gossip protocols are in the areas of blockchain and federated learning [5]. In the first case, the gossip protocol is typically used to update neighbors in the blockchain. The randomized

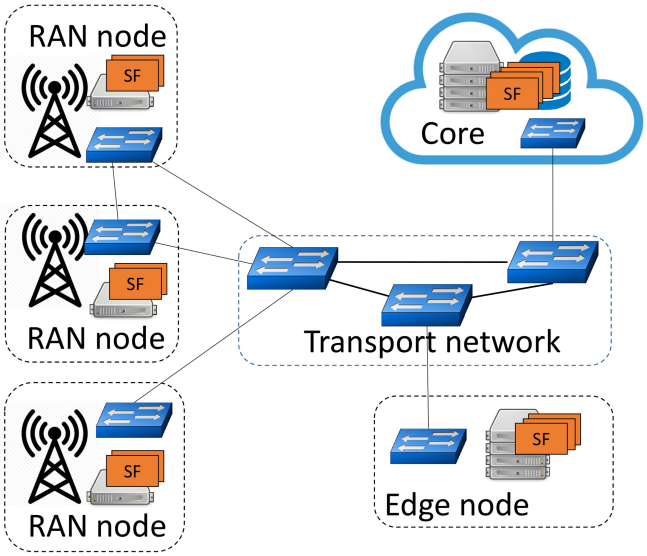


Fig. 1. Considered 6G network scenario.

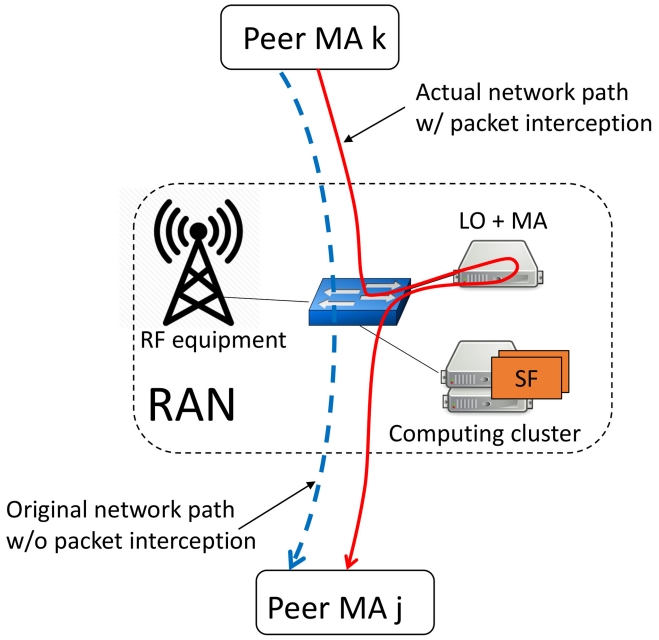


Fig. 2. Packet interception of a gossip session at a RAN node between MA k and MA j.

selection on which gossip is based makes the overall process sustainable, and avoids sending data to all neighbors. Thus, gossip protocols are mainly used as consensus protocols between neighbors [11], [34], [35]. In addition to blockchain-specific proposals, the distributed nature of modern computing paradigms, such as in 6G, has stimulated a revival of gossip as (i) randomized consensus protocol in more general contexts promoting fundamental research [36], [37], [38], and (ii) technology to efficiently monitor systems on a large scale [39].

Finally, gossip is emerging as candidate messaging technology for enabling the convergence of training process in distributed AI applications. For example, gossip is used to efficiently exchange model information between computing clusters [10], [40], [41]. In fact, centralized ML algorithms, adopting stochastic gradient descent, may suffer from variable latency. The decentralized and asynchronous nature of gossip can successfully address this issue [10], [41].

III. Gossip-Based Monitoring Function

We consider the 6G system architecture sketched in Fig. 1, where the wide-area cloud spans from the radio segment to the core [4]. Each computing cluster includes an LO, with its own monitoring agent, the MA, as depicted in Fig. 2. The MA is responsible to query the SF instances running in the local cluster to retrieve their service status (e.g., SF type, maximum capacity, current load, relevant slice), as well as information about compute cluster status. In addition, each MA exchanges its information with other peer MAs to distribute the service status of the controlled cluster all over the overlay management network so as to provide a global monitoring service, eventually differentiated per slice [3]. A specific feature of the proposed architecture is that each MA has a tree-like view of the network.

In this view, each cluster is represented by the relevant MA (see Fig. 1), and each link in this tree is labeled with the IP distance between the MAs and their estimated communication latency. Gossip packets exchanged between peer MAs are intercepted by other MAs that *lay* on the path between them by means of packet interception, realized by network softwarization techniques available in the cluster, as shown in Fig. 2. The proposed solution is based on a gossip-based discovery protocol that carries in the message payload also the monitoring data. Thus, by executing the mutual discovery, MA entities also update the status information of their peers.

The approach used to discover MA nodes consists of a gossip protocol [42], leveraging SDN packet interception. Gossip sessions are round-based and asynchronous. The period of each round is set equal to T_{gossip}, which is a design parameter. These sessions are established between two nodes, an *initiator* and a *responder*, through a three-way handshake, consisting of three messages: Registration, Response, and Ack. At the beginning of each round, the *initiator* sends a Registration message to the *responder*. When the *responder* receives this message, it replies with a Response. The handshake is closed by a final Ack message sent by the *initiator*. The final Ack is needed to acknowledge the data carried in the Response message, which in turn acknowledges those included in the Registration. As in other gossip protocols (e.g., see [29]), both the Registration and the Response messages include a list of (MA) peers that the *initiator* and the *responder* may want to share with each other, referred to as peers to share (PTS). The set of PTS included in a message is called $List_{PTS}$. Therefore, each node can establish gossip sessions with other (possibly unknown) nodes on subsequent rounds.

Differently from other gossip protocols, the SDN packet interception capabilities allows the Registration message to be received and processed not only by the *responder*, but also by the MA nodes on (or close to) the path from the *initiator* to the *responder* (see Fig. 2). We call these intermediate, intercepting MA nodes *forwarders*. The whole procedure is

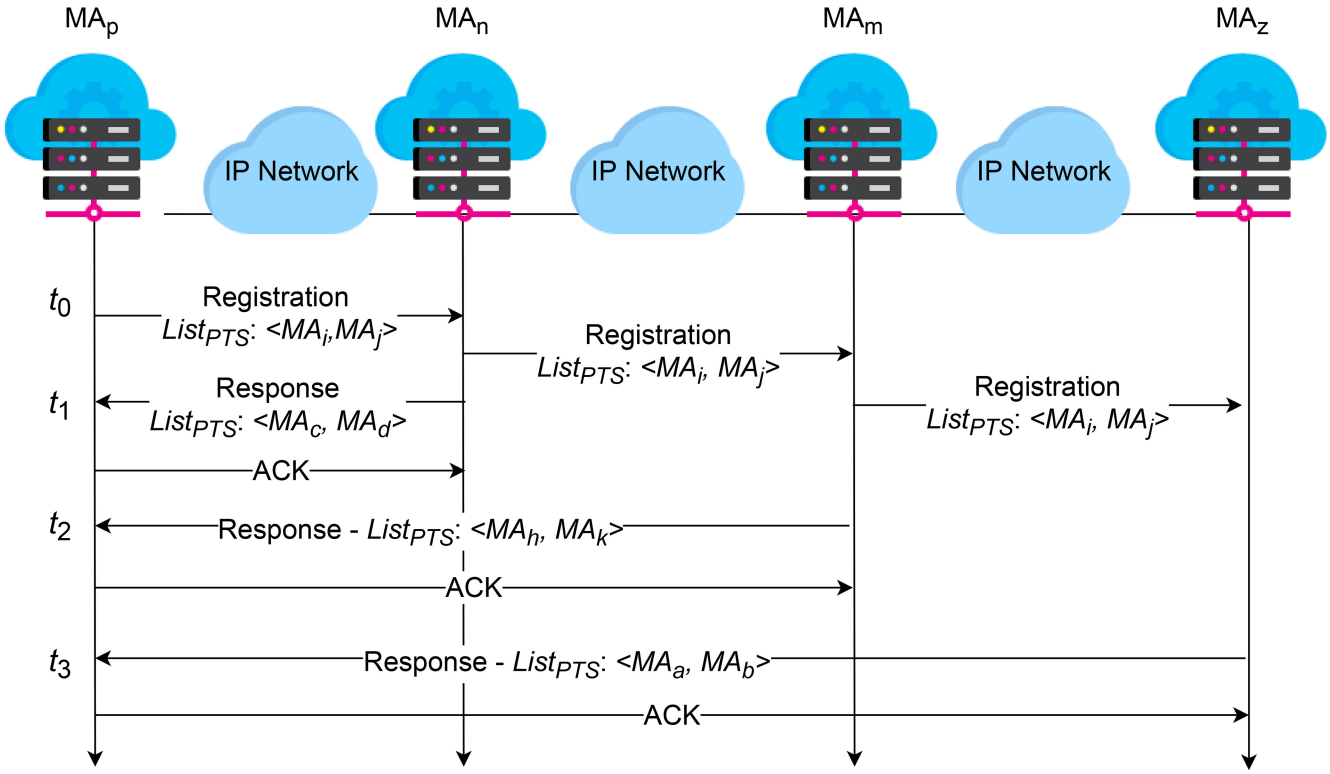


Fig. 3. Gossip discovery of MA entities enhanced with SDN-based packet interception: gossip session and path discovery from MA_p to the MA_z; MA_n and MA_m act as forwarders.

illustrated in Fig. 3. Thus, these nodes actively participate to the discovery process by sending Response messages towards the initiator, sharing their own $List_{PTS}$. In addition to send a Response message, an intercepting MA forwards the message towards the original destination (*responder*). In this way, with a single Registration message, the *initiator* provides an update of the status of its monitored resources not only to the *responder*, but also to all *intermediate* MA nodes. We assume that the system is configured to intercept just the Registration messages, and not Responses and Acks (*default* configuration). However, this assumption can be relaxed and it could be possible to *passively* intercept also Responses, without triggering any other action than acquiring additional information. In fact, by using this second option, referred to as *option 2*, it is possible to report status information in the Response messages from both *responder* and *forwarders*, so they can timely disseminate their status information not only to the *initiator*, but also on the intermediate nodes on the reverse path from *responder/forwarders* to it. With reference to Fig. 3, this means that the information sent by MA_m (both $List_{PTS}$ and monitoring data) would be read also by MA_n, whereas that sent by MA_z would be shared also with MA_n and MA_m. The full implication of this option with respect to the *default* configuration will be discussed in Section IV. Anyway, it is clear that this second option would facilitate the distribution of a larger number of MA identities in each gossip round.

By handshaking with each node, the *initiator* can evaluate its downstream distance from all MA nodes along the path, in terms of both MA hops and, roughly, latency for reaching each of them. Thus, this procedure aims at discovering overlay paths and evaluate the associated metrics, so as to allow each initiator building a tree-like view of the MA network.

When a MA node is turned on, its list of reachable MA nodes is empty. Thus, it is necessary to statically configure the address of an always-on node, called *tracker*, so as to have at least a MA to gossip with. In other words, the *tracker* acts as the first MA *responder* (node on the right-hand side of Fig. 3). A reasonable choice is to use the MA co-located with the CO, or another node in a central position in the operator network, as tracker.

After this initial procedure, a MA node knows (at least) one additional MA node (i.e., the one communicated in

TABLE I
NOTATION AND ABBREVIATIONS

Acronym	Meaning
CO	Central orchestrator
LO	Local orchestrator
MA	Monitoring agent
SDN	Software-defined networking
NFV	Network function virtualization
SF	Service fucntion
PTS	Peer-identity to share in a gossip message
$List_{PTS}$	List of PTS
PE	Peer element, the identity of another discovered MA
PeT	Peer table, storing discovered PEs
pathList	Overlay network path made of a sequence of PEs
LP	Leaf peers: the PE at the end of a *pathList*
PaT	Path table, storing *pathLists*
PTG	Peer to gossip
peerList	Temporary path list
T_{gossip}	Period of a gossip round
T_{cycle}	Period of information distribution to all MAs in the overlay

the Response message from the *tracker*), and can periodically establish a gossip session with it, in order to update its $List_{PTS}$ and exchange monitoring information. Clearly, each MA cannot know when it has discovered all the other MA nodes in the network (end of *discovery phase*), since it would require the *a priori* knowledge of the whole network, which we want to avoid. In fact, network topology can be dynamic. Thus, the distinction between *discovery* and *steady state* phase is artificial, and useful for only for performance evaluation. However, from the protocol viewpoint this not a critical issue. In fact, the discovery protocols has also the function of exchanging monitoring data, thus, when discovery completes, the second function will continue running anyway.

A. Mathematical Model

Before entering the full protocol details, we present a mathematical model of the proposed gossip protocol, to better understanding our design choices. The network is modeled as a graph of MA nodes, referred to as MA overlay, denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. $\mathcal{V}$ is the set of nodes with cardinality $K = |\mathcal{V}|$, and $\mathcal{E}$ is the set of the undirected edges. The routing of gossip packets, which determines the elements in $\mathcal{E}$ connecting MA peers, uses the underlying IP routing, adopting the shortest path policy. As already mentioned above, MAs intercept Registration packets. We define a path $\pi_{ij} = \{i, k, \ldots, j\}$ as the ordered sequence of MA nodes on the network path from i to j, and we denote by $s_{ij} = \pi_{ij} - \{i\}$ the path without the source node, that is the sequence of MA nodes visited by a packet sent by the peer i towards the peer j. We define $\mathcal{S} = \{s_{ij} | i, j \in \mathcal{V}\}$.

We now focus on the discovery phase. It allows all MA nodes in $\mathcal{V}$ to receive the identities of the other MA nodes and to evaluate the relevant metrics, which requires exchanging messages with all the other MAs. Since the protocol is round-based, the minimization of the discovery time translates in minimizing the number of gossip sessions. We model this problem as a *set covering problem* (which is a class of problems known to be NP-hard, see [43]): given a node $i \in \mathcal{V}$ and the associated universe $\mathcal{U}_i = \mathcal{V} - \{i\}$, the set $\mathcal{S}_i = \{s_{ik}, k \in \mathcal{D}_i \subseteq \mathcal{U}_i\}$, $\mathcal{S}_i \subseteq \mathcal{S}$, is a cover for $\mathcal{U}_i$ if the union of its elements contains all elements in $\mathcal{U}_i$. Thus, it is possible to formulate the following problem C_1:

$$\min \sum_{i=1}^{K} |\mathcal{D}_i|, \tag{1}$$

subject to

$$\mathcal{U}_i = \bigcup_{s_{ik} \in \mathcal{S}_i} s_{ik} = \bigcup_{k \in \mathcal{D}_i} s_{ik}, \forall i \in \mathcal{V}. \tag{2}$$

The solution of this problem, that is the identification of the minimum sets $\mathcal{D}_i, \forall i \in \mathcal{V}$, provides a solution of the discovery problem for all MA peers. In fact, $\mathcal{D}_i$ is the minimum set of peer MAs to gossip from node i to contact all the other MA nodes in $\mathcal{V}$ by leveraging packet interception capabilities of the system. This means also that $|\mathcal{D}_i|$ is the minimum number of gossip rounds necessary to the node i to complete network discovery. For each MA node $i \in \mathcal{V}$, we define the single-source shortest-path tree $\mathcal{T}_i$ rooted at i, [43]. $\mathcal{T}_i$ identifies the MA nodes on the (shortest) path from i towards any other node $k \in \mathcal{V}$. An example of $\mathcal{T}_i$ for a very simple graph $\mathcal{G}$ is drawn in bold in Fig. 4, where $i = 1$. We say that a node $h \in \mathcal{V}$ is a leaf for i if it is a leaf for the tree $\mathcal{T}_i$. We denote as $\mathcal{L}_i$ the set of leaf nodes for i, and $M_i = |\mathcal{L}_i|$, thus from the leaf definition it results that if $h \in \mathcal{L}_i$, then $h \in s_{ij} \Leftrightarrow h = j$. Paths associated to leaves for node 1 are shown by red dashed arrows in Fig. 4.

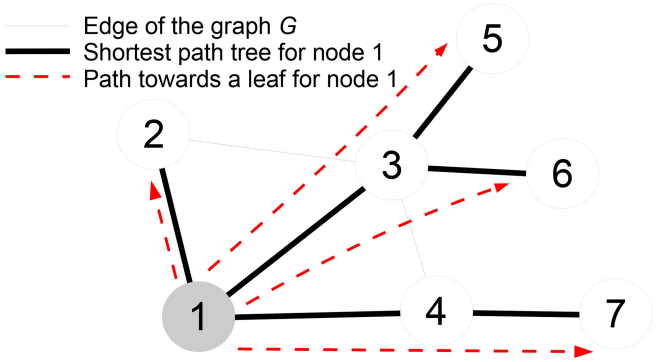


Fig. 4. Example of a possible subset of the universe and solution for the discovery problem for node 1. The tree $\mathcal{T}_1$ is drawn in bold.

Our proposed solution of the problem C_1 is based on the following consideration. If a node i executes a gossip session with all the leaves of its $\mathcal{T}_i$ tree, it certainly discovers all the (MA) nodes in $\mathcal{G}$, together with the relevant metrics, thanks to interception of Registration messages. Thus, our solution is aimed to quickly discover all Leaf Peers (LPs) of the tree associated with each node in $\mathcal{V}$.

Theorem 1: The optimal solution $\mathcal{D}_i^*$ to the set cover problem C_1 is given by the sets of leaves for each node in the overlay, that is $\mathcal{D}_i^* = \mathcal{L}_i, i \in \mathcal{V}$.

Proof: By definition of leaf, each node $h \in \mathcal{L}_i$, $i \in \mathcal{V}$, belongs to the optimal set of solution $\mathcal{D}_i^*$, that is $\mathcal{L}_i \subseteq \mathcal{D}_i^*$, otherwise $\mathcal{U}_i$ is *not covered*. We show that $\mathcal{D}_i^* = \mathcal{L}_i$. Assume, by contradiction, that $\exists z \in \mathcal{D}_i^* - \mathcal{L}_i$. Then, since z is not a leaf, $\exists y \in \mathcal{L}_i \mid z \in s_{iy}$. Thus, from the shortest path routing assumption, it follows that $s_{iz} \subset s_{iy}$. Since $y \in \mathcal{L}_i \subseteq \mathcal{D}_i^*$, thus $\mathcal{D}_i^* - \{z\}$ is still a solution of C_1 (see (2)) but with a lower cost than $\mathcal{D}_i^*$ (see (1)). Consequently $\mathcal{D}_i^*$ cannot be an optimal solution for i. ■

Similarly, it is easy to show that $\mathcal{L}_i$ is the solution also for the following optimization problem C_2 modeling network overhead:

$$\min \sum_{i=1}^{K} \sum_{j \in \mathcal{B}_i} \left[p_{ij}\, q + \sum_{z \in s_{ij}} (p_{zi}\, r + p_{iz}\, a) \right], \tag{3}$$

subject to

$$\mathcal{U}_i = \bigcup_{k \in \mathcal{B}_i} s_{ik}, \forall i \in \mathcal{V}, \tag{4}$$

that is $C_2 \equiv C_1$, since they have the same solution $\mathcal{B}_i = \mathcal{D}_i = \mathcal{L}_i$, where

- p_{ij} is the distance from the node i to the node j on the tree $\mathcal{T}_i$, i.e., $p_{ij} = |s_{ij}|$.

- *q*, *r*, and *a* are the size of the Registration, Response, and Ack messages, respectively (Fig. 3).

B. The Peer Selection Algorithms

The main outcome of Section III-A is that the optimal set of peer MAs to be gossiped, in order to minimize the completion time and the overhead of the discovery of all MAs in the network, is represented by the set of LPs. However, as already explained, for robustness and simplicity, we provide neither the topology nor the set of MAs participating in the monitoring process. Thus, although the optimal solution of the network discovery problem is known, it is difficult to implement it in practice, since the system works in a distributed fashion with an incomplete knowledge of the overlay, which increases step by step. In addition, it is a circular problem: the identities of MA nodes are not known at bootstrap, since they are discovered during the execution of the network discovery, which we want to optimize by contacting only specific nodes (often still unknown), i.e., the LPs, to limit the number of necessary gossip rounds, as in Fig. 3.

Thus, we need to design an heuristic procedure to quickly discover the LPs of each MA node, since all the other MA nodes will be discovered by intercepting Registration messages. For this reason, we call this solution *Leaf-based*.

A tricky point is that each MA, in order to let other MAs quickly discover *their* LPs, should exchange only the identities of *potential* leaves in the $List_{PTS}$ field of Registration and Response messages (Fig. 3). To this aim, we have defined a couple of simple, lightweight, and soft-state structures storing peer information at each MA node.

- The former, called *peer table* (PeT), stores the identities of the other MA peers together with their associated metrics (peer element, PE).
- The latter, called *path table* (PaT), stores in MA_i the set of overlay paths, *pathLists*, with *i* as first node, i.e., the *ordered* sequence of PEs in the set containing s_{ij}, as new MA nodes are discovered and contacted.

The PeT is computed by each node receiving a message, both Registrations or Responses, carrying a previously unknown PE as *initiator* or in the $List_{PTS}$. Instead, the PaT is computed by each *initiator i* by inspecting the Response messages sent by any intermediate node *k* that has intercepted the Registration message destined to a *responder j*, as shown in Fig. 3. In this regard, we point out that a node *z*, which is a leaf for *i*, is not necessarily a leaf also for *j*, which receives *z* in the $List_{PTS}$ sent by *i*. In addition, it may happen, especially in the initial rounds, that a newly activated MA node knows just a limited number of peers, thus the identities it shares could not be *true* leaves.

Two algorithms are executed in MA nodes, since the *initiator* has to *select* two types of peers stored in the PeT:

- the so-called *peer to gossip* (PTG), which is the intended recipient of the message, i.e., the *responder*,
- the $List_{PTS}$ to insert in the Registration message; this list includes the PE identities to share with the PTG and any intercepting MA node.

Before delving into the detail of these two algorithms, it is necessary to explain the operations carried out when the *initiator* receives the Response sent by a remote peer:

- The *initiator* adds each element of the received $List_{PTS}$ not already present in the PeT as new PE, with flags <*isGossiped*, *isContacted*> set to <true, false>. This is important for subsequent selection of PTG and PTS. In fact, since each node tries to share just LPs, an identity received in a $List_{PTS}$ is a good candidate for being selected as future PTG, if all participating peers adopt the same strategy.
- If not already present, the *initiator* adds each intercepting MA to the PeT together with its metrics, and the relevant flag *isContacted* is set to "true". This peer is not a good candidate for future selection of the PTG or to share as a PTS element, since it is not an LP for the *initiator*, being intercepted during a gossip round.

In addition to any other ancillary fields, needed for correct protocol operation, such as header size, version, and so on, the header of a gossip message has to include necessarily the following fields:

- Type of message: Registration, Response, Ack.
- *Initiator* identity: routable IP address.
- *Responder* identity (i.e., the PTG): routable IP address.
- Distance: expressed in term of MA hops. In Registration, this field is initialized to 1 and then has to be increased by each MA intercepting the message before forwarding it downstream, whereas in Response messages this field has to be reported as received.
- The $List_{PTS}$: list of PTS identities, whose size is *H*, which is another field of the protocol header.
- Session Id: it is used to identify univocally a gossip session, including all messages exchanged with *responder* and *forwarders*.

PE identities can be complemented also with unique PE identifiers (PE_Ids), if deemed useful for indexing purposes in internal data structures. In addition, Registration messages have also a payload, consisting of monitoring data. Since we target UDP as transport protocol for its lightweight, connectionless operation, we recommend the usage of (compressed) JSON to encode these data and fit them into a single message.[1] We can also consider a variant of this approach, in which also Response messages may optionally carry a payload containing updated monitoring data from *responder* and *forwarders*. In this case, it could be convenient to allow interception of Responses as well, so that also intermediate MA in the reverse path from forwarders to initiator can benefit from updated information. We analyze the performance impact of this option (*option 2*) in Section IV.

The *initiator* updates a temporary path list (*peerList*) as it receives Responses from intermediate *forwarders*. The position in the *peerList* of a peer is exactly its distance in MA hops. The procedure is completed when the MA hop distance of the PTG is equal to the number of received responses (size

[1]It is also possible to establish of long-lived TCP sessions between peers over which carry out monitoring data exchange, but this option does not change significantly the overall protocol and its performance.

TABLE II
EVOLUTION OF DATA STRUCTURES IN MA_p AS FUNCTION OF EVENTS DURING THE GOSSIP SESSION IN FIG. 3. NEW ENTRIES OR CHANGES AT EACH EVENT ARE HIGHLIGHTED WITH BLUE COLOUR

Event	**Peer Table (PeT)**			**peerList**	**Priority lists for PTG selection**		
time	PE_id	<isGossiped>	<isContacted>		high_priority	low_priority	no_priority
t_0	MA_z	True	False	$\emptyset$	<MA_z,MA_i,MA_j>	<MA_t>	$\emptyset$
	MA_i	True	False				
	MA_j	True	False				
t_1	MA_z	True	False	<MA_n>	<MA_z,MA_i,MA_j,MA_c,MA_d>	<MA_t>	$\emptyset$
	MA_i	True	False				
	MA_j	True	False				
	MA_n	False	True				
	MA_c	True	False				
	MA_d	True	False				
t_2	MA_z	True	False	<MA_n,MA_m>	<MA_z,MA_i,MA_j,MA_c,MA_d,MA_h,MA_k>	<MA_t>	$\emptyset$
	MA_i	True	False				
	MA_j	True	False				
	MA_n	False	True				
	MA_c	True	False				
	MA_d	True	False				
	MA_m	False	True				
	MA_k	True	False				
	MA_h	True	False				
t_3	MA_z	True	True	<MA_n,MA_m,MA_z>	<MA_i,MA_j,MA_a,MA_b,MA_c,MA_d,MA_h,MA_k>	<MA_t>	<MA_z>
	MA_i	True	False				
	MA_j	True	False				
	MA_n	False	True				
	MA_c	True	False				
	MA_d	True	False				
	MA_m	False	True				
	MA_k	True	False				
	MA_h	True	False				
	MA_a	True	False				
	MA_b	True	False				

of the *peerList*), and the last element of that list is exactly the PTG. If this condition is not met, at T_{gossip} expiration, the path is truncated at the last peer having a position equal to its distance. In any case, the new path is added to the *pathList* stored in the PaT. Finally, the *initiator* sends an Ack message back, which does not include any PTS. This allows any *forwarder* or the *responder* to be sure that the *initiator* has been reached by its Response, and thus duly updates the relevant flag in the local data structure. In addition, through this procedure, the *initiator* can also roughly estimate the round trip latency of any responding peer.

Having explained how new PE identities are managed in data structures handled by MAs, we now focus on the selection of PTS elements. The selection of PTS elements is a common process to *initiator*, *forwarders*, and *responder*. Since the Leaf-based gossip protocol aims at gossiping LPs, and shared identities are good candidates to be gossiped, if the PaT includes at least H paths, H randomly selected LPs of these paths are used. Otherwise, the node tries to fill the $List_{PTS}$ by using peers already discovered but still not contacted, that are identifiable by the flag *isContacted* $=$ *false* in their PEs. Such peers are those that have already gossiped the selecting node, which has acted as *responder*, or those whose identities have been shared by other nodes (i.e., they have also the flag *isGossiped* $=$ *true*). Since uncontacted peers might also be LPs, this approach is preferable with respect to use of peers already contacted, which are not LPs, given that all nodes should preferably share LPs. After a peer has been contacted (as either a *responder* or *forwarder*), its *isContacted* flag is updated to *true*. Thus, it would be a candidate for being shared as PTS only if it is a LP for a path.

The other selection algorithm is the one used to determine the PTG, randomly picked up from three priority lists. The first list, referred to as *high priority*, includes uncontacted PEs with the flag *isGossiped* $=$ *true*, since they are most likely LPs. The second list, referred to as *low priority*, includes uncontacted PEs with the flag *isGossiped* $=$ *false*, that are not likely to be LP. Those peers have gossiped the current MA, but not vice versa. Finally, the third list, referred to as *no priority*, includes all LPs of the PaT. Thus, uncontacted peers are preferably selected, in order to quickly accomplish network discovery. When all peers have been contacted (priority lists are empty), peers enter the steady phase, during which just LPs are gossiped, in order to update the status of the highest possible number of peers with a single Registration message.

The PaT consistency is guaranteed by updates done when a new path is collected during a gossip session. The result of a gossip session can imply merging, updating, or truncating paths already present in the *pathList*, especially during the initial building of the PaT. These procedures are cumbersome to describe but straightforward in their operation, thus they are not detailed here.

In order to better understand how the proposed procedures work together, we resort to an operational example. Table II shows the evolution of information stored in *initiator* at the reception of different messages from *forwarders* and *responder* during the exchange depicted in Fig. 3. The configuration at the beginning of the gossip round is that reported at t_0. At this time, MA_p, which is the *initiator*, selects MA_z as *responder* from the *high_priority* list and sends a Registration to it. Since MA_n intercepts that message, it replies with a Response, including in the header the identities of peers MA_c

and MA_d as PTS. At t_1, three MAs are added to the PeT: MA_c and MA_d are gossiped by MA_n, thus they are inserted in the *high_priority* list, whereas MA_n is added to the transient *peerList*. Note that the flags associated to these nodes are different, as per the described algorithms: for MA_i and MA_j, the flags are set to *isGossiped=true* and *isContacted=false*, whereas for MA_n, which is a *forwarder*, they are opposite, since it is contacted (by interception) by the *initiator* and it was not previously gossiped by another MA. The process repeats for MA_m (another *forwarder*) and MA_h and MA_k (shared peers by MA_m) at t_2. Finally, when the Response of the PTG (MA_z) arrives, the *peerList* is completed and uploaded to the PaT, whereas MA_z is moved from the *high_priority* list to the *no_priority* one, with its flag *isContacted* updated to *true.*

An interesting by-product feature of the proposed approach is the following. The steady state is reached when each MA has all its LPs listed under the *no_priority* list, with the *high_priority* and *low_priority* lists empty. However, upon a new MA enters the system and gossips the *tracker*, its identity will start being shared. This favours its dissemination in one of the two transient list, thus accelerating the mutual discovery with all the other peers and thus the reaching of a new steady state condition.

Finally, it is worth highlighting that, since PE states are soft, they are removed if not refreshed. The lifetime value is local and depends on the number of paths in the PaT (*pathList*) of each node *i*, which is equal to the number of leaves M_i. Let us define $M_{leaf} = \max_{i \in \mathcal{V}} \{M_i\}$. Thus this lifetime is set equal to $T_{gossip} \times M_{leaf} \times (1 + \Delta)$, where Δ is a margin used to avoid accidental PE cancellations. A safe measure is to gossip a leaf before cancellation at the expiration of its lifetime. Cancellation can happen in case of lack of answer.

C. Implementation Issues

The implementation of the proposed system is very well suited to cloud-native technologies. In particular, by using a cluster orchestrated by Kubernetes (K8s),[2] the MA can be implemented as a function running in a pod of the cluster. This way it can retrieve the cluster status by querying the Kubernetes API, and leverage network softwarization capabilities of K8s. Hence, it is possible to implement a multi-cluster, distributed monitoring. Indeed, it would also be possible to integrate this proposal with solutions already offering a multi-cluster K8s environment, such as Kubernetes Armada (Karmada).[3] In this case, the proposed MA should be designed to transparently interact with the function offering the Global Uniform Resource View. This includes the interaction with the server, which offers a REST API endpoint, and the data distribution function by using gossiping.

In addition, it is also interesting to consider possible implementation in a multi-cloud environment based on OpenStack.[4] In this case it results even more suited, as shown by a previous research related to deploying a set of OpenStack tenants, running a genomic processing service, and using the OSP protocol [32] for data distribution. Also in this case, an agent was used to query OpenStack APIs to obtain the status of resources per tenant. OSP was used to distribute this information between agents. The results of this experimental campaign are illustrated in [44].

[2]https://kubernetes.io/

[3]https://karmada.io/

[4]https://www.openstack.org/

IV. Performance Evaluation

The performance of our proposal has been evaluated by means of both simulations and theoretical models. The simulation setup, shown in Fig. 5, consists of 60 nodes, which form the *underlay* network over which the gossip *overlay* is built. Both stub and core nodes are included. In particular, 36 network stubs model MAs running in RAN nodes. An additional MA stub node acts as Tracker. Each stub is connected to one of the 23 core nodes, which represent the edge/core MA nodes of the 6G wide-area cloud. The simulation was implemented in MATLAB.[5] In the simulation scripts, each node indicates a cluster with its own MA, whereas the network graphs are represented through sparse matrices. The duration of each simulation run, when executed on a notebook equipped with an Intel i7-1255U processor (10 CPU cores and 12 threads) and with 16 GB of RAM, is about 2 minutes.

We evaluated the performance of our solution versus literature counterparts with two different *overlay* topologies:

- *Full topology:* All 60 nodes include co-located computing clusters and run MA instances. In this configuration, the gossip *overlay* network corresponds to the physical *underlay* network.
- *Partial topology:* 48 nodes out of 60 - 80% - have computing clusters associated with the relevant MAs. They are 36 stubs, 11 core, and the tracker. This means that only these nodes constitutes the gossip *overlay*, whereas 12 core nodes act as standard IP routers (namely C11-C22), with only transport functions in the *underlay* network. These nodes and relevant links are consequently transparent for the *overlay*, which becomes more meshed.

Clearly, the partial topology is the most realistic one for two reasons. The first is that likely not all nodes in a future mobile network need to have a co-located computing cluster (edge cloud), and can be simply IP transport nodes. The second is that the partial topology models an incremental scenario, in which computing clusters are gradually added to some nodes in key positions on the overall underlay topology.

As for the performance evaluation, we compared the results of our proposed, fully distributed approach (labeled as "Leaf" in what follows) with alternatives taken from the literature:

- The OSP protocol [32], which was specifically designed to distribute signaling traffic (monitoring information in this scenario) in virtualized architectures.
- A gossip solution with a pure random peer sampling, labeled *Random*, used since it is a typical choice in the gossip literature (e.g., see [45]).
- A *centralized publish-subscribe* monitoring architecture taken from the recent literature, such as the one based on Kafka and described in [22], [23]. This pub/sub solution allows distributing the information to all interested

[5]https://it.mathworks.com/products/matlab.html

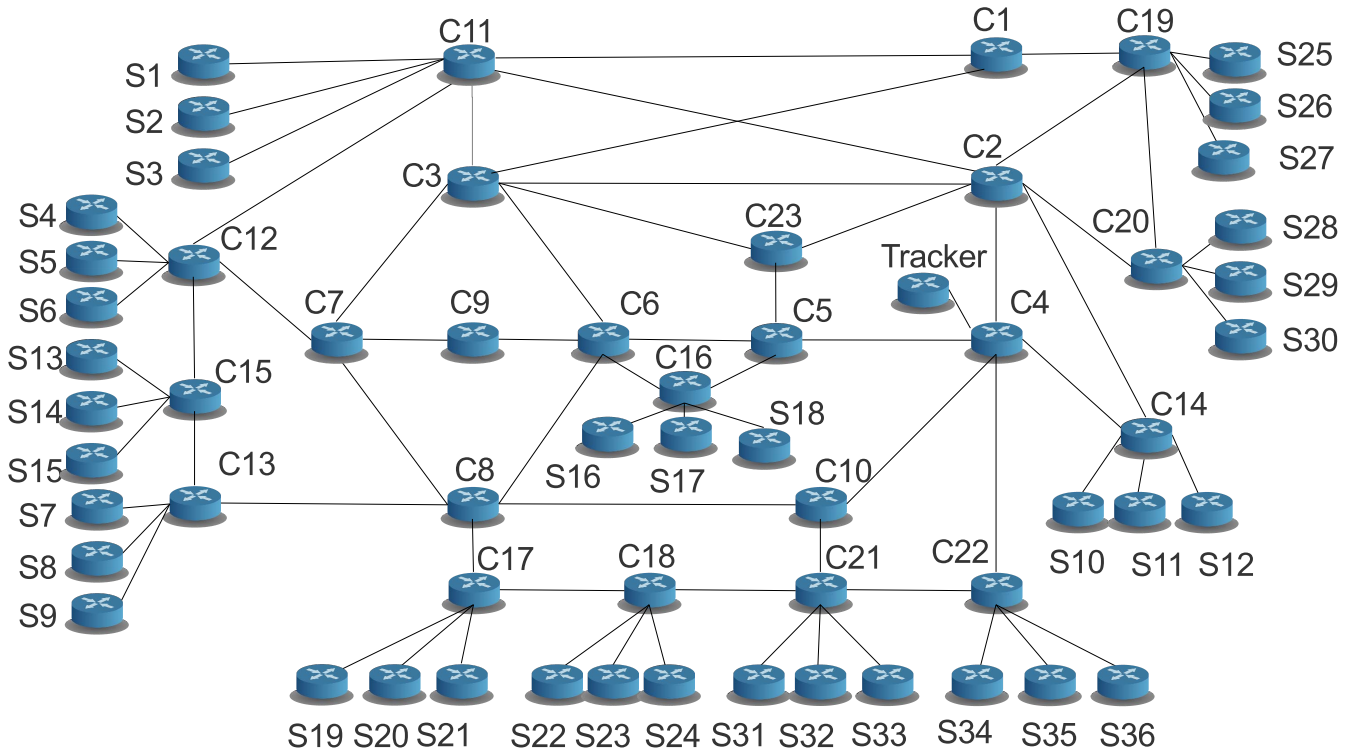


Fig. 5. Network topology (*undelay*): S1-S36 indicate stub MAs, C1-C22 core MAs.

entities of the 6G system (so it supports distributed control and decision layers), but such a distribution makes use of a *central broker*. Without any loss of generality, we used the realistic assumption that the *central broker* is co-located with the Tracker in the topology of Fig. 5.

For these solutions, we evaluated two key performance indicators (KPIs):

- The discovery/convergence time for distributed solutions (Leaf, OSP, Random), reported in Section IV-A. In fact, in the centralized pub/sub solution, each MA publishes its data directly to the central broker, and each new MA can immediately receive the monitoring data published by all the other MAs by subscribing to the relevant topics managed by the broker. Thus, no transient is present.
- The overhead of the protocols used to distribute the monitoring data in the steady state (Leaf, OSP, centralized pub/sub), reported in Section IV-C. For this KPI, we have not included the Random approach, given its very low performance exhibited in terms of convergence time. In Section IV-B, we present a theoretical model for the overhead of the Leaf approach, validated in Section IV-C.

A. Network Discovery of Monitoring Agents

For distributed solutions, we define the convergence time $t_{conv,i}$ of the node MA_i as the time taken for completing the discovery of all other involved MA peers (MA discovery). In our Leaf solution, as well as in the Random one, we define $t_{conv,i}$ as the time necessary for each node to have at least one complete gossip session with all MA nodes, being them contacted as either *forwarders* or *responders*. Hence, the network convergence time is $T_{conv} = \max_{i \in \mathcal{V}}\{t_{conv,i}\}$. Instead, in the OSP protocol [32], we define T_{conv} as the time needed for each MA to contact all its neighboring MAs, i.e., the peers at distance 1 hop on the overlay topology. In both cases, T_{conv} represents the time needed to complete the transient phase, in which the overlay topology (or the set of adjacent neighbours for OSP) is discovered.

Fig. 6 shows the discovery time T_{conv} as a function of H, the number of shared PEs identities in a gossip message, reported on the abscissa axis, for both the full topology Fig. 6(a) and the partial one Fig. 6(b). In these experiments, we normalized the discovery time to the duration of the gossip period T_{gossip}, thus it is expressed as number of gossip rounds and not in seconds, so as to make results more general and not tied to the specific scenario. For all curves, we plot 95% confidence intervals.

In the full topology case, both Leaf and OSP solutions provide satisfactory convergence time, whereas the one of Random is about one order of magnitude larger. To take into account this difference, we used the logarithmic scale on the ordinate axis. It emerges that, when the Leaf solution is used, the convergence time is mostly stable, as witnessed by the very small confidence intervals, and nearly constant for $H \geq 2$. This result is valid in general, since sharing 2 identities is enough to provide each peer with a sufficient number of "uncontacted" peers, which could fill the *high_priority* list in its PeT, as shown in Section III-B and Table II therein. In this way, uncontacted peers can be selected as a PTG in subsequent gossip rounds. By using the OSP solution, the convergence time increases by the number of shared identities H. In fact, sharing many peer identities makes the set of PEs, selectable

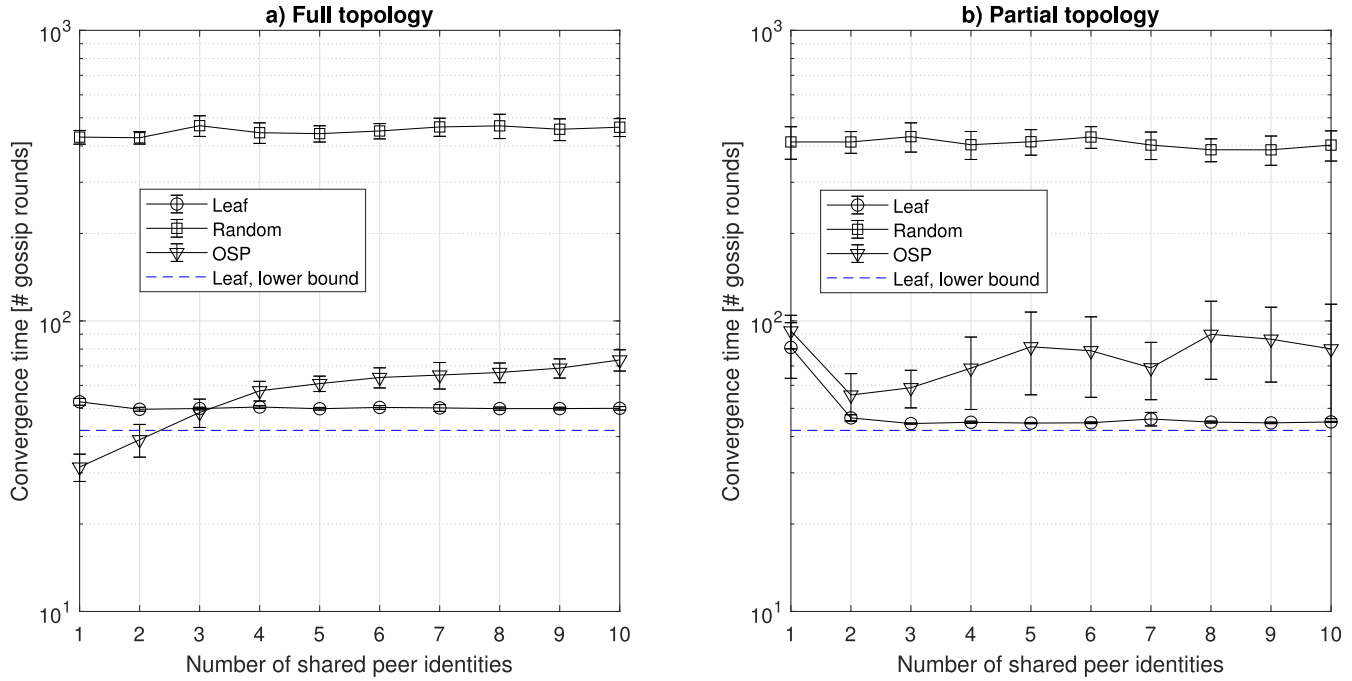


Fig. 6. Convergence time vs. H, the size of the $List_{PTS}$: a) full topology, and b) partial topology; figure reports also 95% confidence intervals.

as the next PTG, large. Since in this way the number of possible PTGs is much larger than the number of MA neighbors (i.e., those adjacent on the overlay, which are the target of OSP gossiping), for the OSP protocol it is disadvantageous to test a large set of PEs, most of which are unreachable at MA level (see also [32] and relevant supporting document).

For the full overlay topology, we observe that when the number of shared peer identities is 1 or 2, this phenomenon does not emerge and the OSP is preferable to Leaf, being lower the number of MAs to discover and having to pick them from a limited set.

Instead, in a sparse, partial topology, the Leaf solution is preferable for all values of H. In fact, in a sparse topology, the discovery time of the OSP solution tends to increase, due to the fact that the average number of "neighbors" on the overlay increase. In addition, its variability results incredibly high, which is not a desired feature. Instead, the discovery time of the Leaf solution slightly decreases for the partial topology, since less MA peers imply less gossip rounds and thus lower discovery time, as expected. This is true also for the Random solution, which however is always 10 times slower in converging. Thus, in the end, the Leaf solution has a much more stable and predictable behavior, and is preferable. Again, the value of $H = 2$ seems to be large enough to speed up the convergence time, without increasing so much the signalling overhead, which will be evaluated in the next section.

Since the Leaf solution is designed to discover leaves, a significant decrease of the convergence time is expected only when the number of leaves decreases, whereas a less important decrease is expected if we remove from the overlay some core nodes. Thus, given the topology overlay depicted in Fig. 5, removing nodes in the range C11-C22 (core nodes) from the overlay should not significantly decrease the number of leaves, most of which are stub nodes of the overlay (i.e., S1-S36). Nevertheless, the net effect is a slight decrease in T_{conv}, as expected, since our heuristic requires less round to converge, having less nodes to test with gossip attempts.

Finally, from the analysis of the two sub-figures Fig. 6a and Fig. 6b together, it emerges that the value of the maximum number of leaves M_{leaf} for the Leaf protocol, in this specific case, is the same for the partial and full topology. It is indicated by the dashed blue in both sub-figures. Since by definition $M_{leaf} \leq T_{conv}/T_{gossip}$, it represents a lower bound for the normalized convergence time of the Leaf solution. We can appreciate that the Leaf solution approaches quite well this lower bound.

B. Monitoring Data Delivery Overhead: Theoretical Models

When all MA nodes are discovered, the goal consists of exchanging the information about the status of computing clusters in the most efficient and quick way. In the steady state, each MA i has the list of its own leaves to gossip, equal to $\mathcal{L}_i$ with size M_i. Consequently, the minimum time needed to complete the distribution of information towards all leaves, and thus to all MAs in the overlay, is equal to $T_{cycle} = M_{leaf}\, T_{gossip} \leq T_{conv}$. In order to correctly evaluate the network overhead of this process, which is computed at IP layer,[6] it is necessary to include in the graph modeling our topology not only MA nodes, but also IP routers. This means that we have to consider a new extended graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$, which models the underlay and clearly includes the nodes of the overlay topology (MAs), that is $\mathcal{V} \subseteq \mathcal{V}'$ with $K = |\mathcal{V}|$, and $\mathcal{E}'$ being the set of undirected edges connecting the elements of $\mathcal{V}'$ (IP nodes, including MAs). Let us define

[6]Even if here we specifically consider the IP layer for evaluating the transport overhead, in a more general view we could evaluate it on an *underlay* topology supporting the *overlay* one.

$K' = |\mathcal{V}'|$ and the ratio between the number of overlay and underlay nodes equal to $\rho = \frac{K}{K'}$.

We denote by δ the IP network diameter. In addition, we denote by ξ_{ij} the probability that a peer i selects an LP $j \in \mathcal{L}_i$ as a PTG. Clearly, in steady state conditions, it results $\xi_{ij} = 1/M_i$. We define the network length (measured at the IP network layer) of a path from MA i to MA j as n_{ij}, whereas the path length in the MAs overlay is $p_{ij} \leq n_{ij}$, with the equality holding for the full topology. Thus, the average IP length of paths in the PaT of MA_i is given by $\mu_i = \sum_{j\in\mathcal{L}_i} \xi_{ij} n_{ij} = 1/M_i \sum_{j\in\mathcal{L}_i} n_{ij}$, where the last equality holds in the regime condition only. Furthermore, if an MA node $k \in \mathcal{V}$ has ordered position z in the overlay path s_{ij}, we define the positioning function $g_{ij}(\cdot)$ returning the identity of the MA node having the zth position on that overlay path, i.e., $k = g_{ij}(z)$, with $j = g_{ij}(p_{ij})$ and $i = g_{ij}(0)$.

In the general case of a partial topology, in which not all considered nodes are MAs, the traffic generated by the Leaf solution during a gossip session between MA nodes i and j is equal to (see also Fig. 3 and (3)):

$$\phi_{ij} = n_{ij}\, q + (r+a) \sum_{z=1}^{p_{ij}} n_{i\, g_{ij}(z)}, \tag{5}$$

where z is the ordered position of a given intermediate MA node k in the path s_{ij}. Thus, for each MA in the path, in (5) we account for the amount of IP hops that the Responses and Acks of *responder* and *forwarders* have to cross. By looking to Fig. 3, it is easy to see that (5) can be rewritten as

$$\begin{aligned}\phi_{ij} &= n_{ij}\, q + (r+a)\big[p_{ij} n_{i\, g_{ij}(1)} + \big(p_{ij}-1\big) n_{g_{ij}(1)\, g_{ij}(2)} \\ &\qquad + \cdots + 1 \cdot n_{g_{ij}(p_{ij}-1)\, g_{ij}(p_{ij})}\big] \\ &= n_{ij}\, q + (r+a) \sum_{z=1}^{p_{ij}} \big[p_{ij} - (z-1)\big] \cdot n_{g_{ij}(z-1)\, g_{ij}(z)},\end{aligned} \tag{6}$$

The last version of (6) uses distances between adjacent MAs $k-1$ and k on the overlay path $i \to j$ equal to $n_{g_{ij}(k-1)\, g_{ij}(k)}$. These distances in the summation can be approximated by their average value, which on the path from i to j is equal to n_{ij}/p_{ij}. This approximation works well especially if their variability is not so large. This means that we can write (6) as

$$\begin{aligned}\phi_{ij} &\approx n_{ij}\, q + (r+a) \sum_{z=1}^{p_{ij}} \big[p_{ij} - (z-1)\big] \frac{n_{ij}}{p_{ij}} \\ &= n_{ij}\, q + (r+a) \frac{n_{ij}\big(p_{ij}+1\big)}{2}.\end{aligned} \tag{7}$$

Thus, the average traffic generated by the ith node in a gossip round is $\phi_i = \sum_{j\in\mathcal{L}_i} \xi_{ij}\phi_{ij}$. Consequently, the total network signaling generated in a gossip round can be computed as

$$\Phi_{Leaf} = \sum_{i=1}^{K} \phi_i = K\left(\mu q + (r+a)\frac{R_{np}+\mu}{2}\right), \tag{8}$$

where $\mu = E[n_{ij}]$ is the average path length towards leaves in IP hops, whereas $R_{np} = E[n_{ij} p_{ij}]$ is the cross-correlation between IP path length and overlay path length. Since they are clearly strongly correlated, thus R_{np} cannot be expressed in a simpler form. From (8), it is immediate to estimate the total volume of traffic exchanged between MA nodes to update each other with the information about the status of computing clusters. In fact, disseminating the status of each cluster to all the network requires a number of gossip rounds equal to the maximum number of leafs seen by an MA, that is

$$\Gamma_{Leaf} = \Phi_{Leaf}\, M_{Leaf}. \tag{9}$$

Note that (9) holds for both default configuration, in which the Response message carries just the PTS list, and for option 2 (see Section III), in which the Response carries also information about the status of monitored cluster. The only difference is in the size of r, since in option 2 it has also a payload and not just the header. However, from a closer look to the features of option 2, it is immediate to deduce that it is possible to reduce the total traffic by half to $\Gamma_{Leaf,opt2} = \Phi_{Leaf} \lceil M_{Leaf}/2 \rceil$. For this purpose, it is necessary that each node, when selecting the PTG, makes a choice among those that have not been contacted (either directly as *initiator* or indirectly as *responder* or *forwarder*) recently.

Finally, it is interesting to see that from (8) it is possible also to easily calculate an upper bound to the network overhead, which provides a coarse estimation to maximum network traffic without knowing in details parameters such as η or μ or their correlation, and making use of just the network diameter δ. In fact, since $\mu \leq \delta$ and $n \leq \delta$, it results that

$$\Phi_{Leaf} \leq K\delta\left(q + (r+a)\frac{\delta+1}{2}\right) = \Phi^{UB}_{Leaf} \tag{10}$$

The signaling rate is found by dividing Φ_{Leaf} or Φ^{UB}_{Leaf} by the gossip period T_{gossip}.

If we consider the full topology, it is easy to see that it is possible to simplify some expressions. In fact, since $n_{ij} = p_{ij}$, the traffic generated on the path $i \to j$ becomes

$$\phi^{full}_{ij} = p_{ij}\, q + (r+a) \sum_{i=1}^{p_{ij}} i = p_{ij}\, q + (r+a) \frac{p_{ij}\big(p_{ij}+1\big)}{2}, \tag{11}$$

thus the average traffic generated by all nodes becomes:

$$\begin{aligned}\Phi^{full}_{Leaf} &= \sum_{i=1}^{K} \sum_{j\in\mathcal{L}_i} \xi_{ij}\phi_{ij} \\ &= K\left(q\mu + \frac{(r+a)}{2}\Big(\mu + \mu^2 + \sigma^2\Big)\right),\end{aligned} \tag{12}$$

where σ is the standard deviation of the random variable modeling the path length $n_{ij} = p_{ij}$.

In order to proceed, it is necessary to know the distribution of path lengths to evaluate σ, assuming that it may be easy to estimate the average value μ. Although its distribution could likely have a bell shape, commonly approximated with a normal distribution, it is also true that in the full topology there are some MAs that have some of their LPs at distance 1 MA, whereas the others are on the opposite part of the network. Thus, we use the working assumption that the mass probability function of $p_{ij}, j \in \mathcal{L}_i$, can be considered uniformly

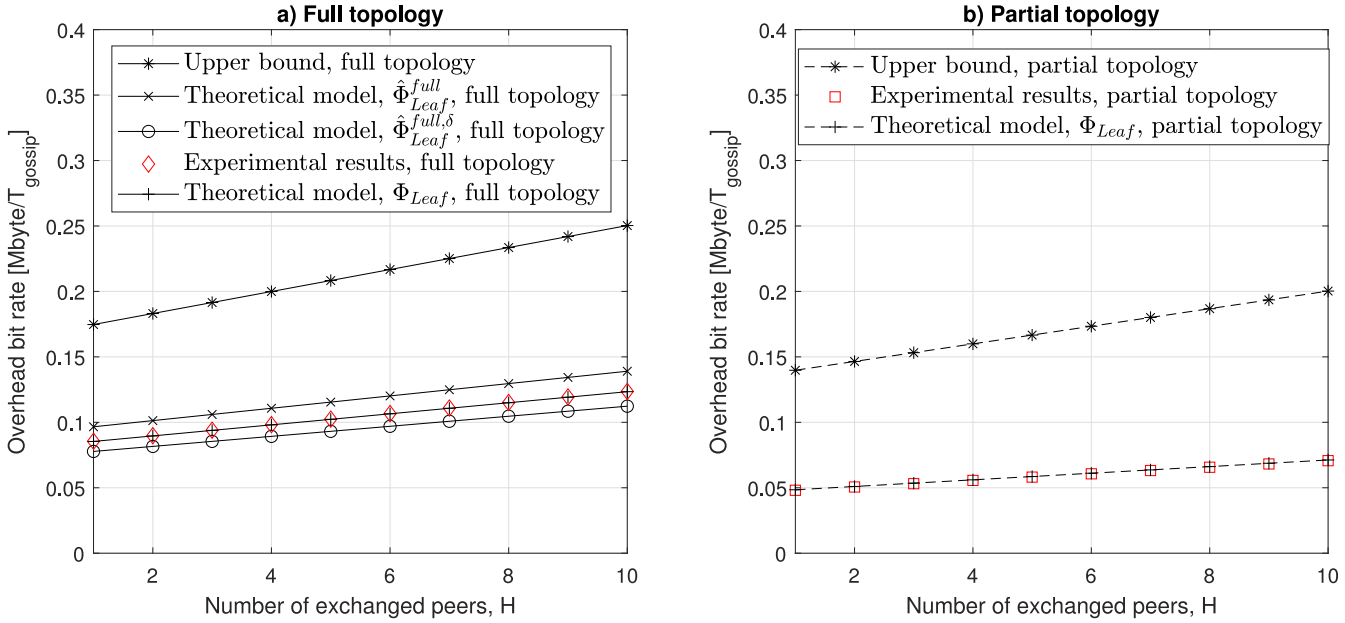


Fig. 7. Overhead per gossip cycle vs. H, the size of the $List_{PTS}$: upper bound, theoretical models, and experimental results for a) full and b) partial topology.

distributed between $p_i^{min} = 1$ and $p_i^{max} \leq \delta$. It is clear that, with such an assumption of flat mass probability, the standard deviation of the approximating uniform distribution $\hat{\sigma} = \frac{\mu^2 - \mu}{3}$ dominates σ, so producing an overestimation of the amount of exchanged traffic (see (12)). Nevertheless, the approach can be still appealing, since it allows expressing $\hat{\sigma}$ as a function of μ, especially in comparison with the upper bound Φ^{UB}_{Leaf} in (10). In fact, from (12) it is simple to show that the total network signaling results:

$$\hat{\Phi}^{full}_{Leaf} = K\mu\left(q + \frac{(r+a)}{3}(1+2\mu)\right). \tag{13}$$

Finally, if we push this approximation further and consider the path length uniformly distributed between 1 and δ at the domain level, we can approximate $\mu \approx \frac{\delta+1}{2}$, so obtaining

$$\hat{\Phi}^{full,\delta}_{Leaf} = K\frac{(\delta+1)}{2}\left(q + \frac{(r+a)}{3}(\delta+2)\right). \tag{14}$$

We do not expect that this last approximation would produce an upper bound. In fact, not necessarily $\mu \leq \frac{\delta+1}{2}$, especially in networks that are not so meshed, thus with an average value μ potentially shifted towards the maximum δ.

C. Monitoring Data Delivery Overhead: Numerical Results

In order to proceed with numerical result, we have to set the values of some parameters. The length of Registration (q) and Response (r) messages is set to 16 bytes, plus 4 bytes for each PEs identity in the $List_{PTS}$ of size H, whereas the length of Acknowledgment (a) is set to 16 bytes. The selected transport protocol is UDP, since protocol reliability and robustness is ensured by gossip. First, we evaluate the amount of the protocol overhead. Thus, we set the payload length $L = 0$ in both q and r. Fig. 7 shows the number of Mbytes exchanged on the network by the Leaf protocol for each gossip cycle (Φ_{Leaf}) as a function of H. We found that the mathematical models closely match the experimental performance, for both full and partial topology. As for the simplified model that uses a uniform distribution for the path length ($\hat{\Phi}^{full}_{Leaf}$), it slightly overestimates the amount of signaling traffic (about 13%), but still provides a very good estimate in the full topology case. As for the upper bounds, in the full topology case it is about 2 times larger than the real values, whereas it increases to about 3 times for the partial topology. This is an expected result. In fact, for the full topology, we approximate $n_{ij} = p_{ij} \leq \delta$ with δ, whereas for the partial topology we approximate not only $n_{ij} \leq \delta$ with δ but also $p_{ij} \leq n_{ij} \leq \delta$ with it, thus with a larger overestimation. Thus, in both cases the upper bound can used just to estimate the order of magnitude of the signaling traffic. Finally, let us comment about the results provided by $\hat{\Phi}^{full,\delta}_{Leaf}$. In this case, since the actual value of $\mu \geq \frac{\delta+1}{2}$, it provides a small underestimation of the signaling traffic. However, it cannot be considered a lower bound in general. In fact, it depends on how close is the approximation on the average path length provided through the diameter to the actual value. If it is close, as in this case, it could provide a good approximation, which is about 10% inferior than the actual traffic.

The overall comment is that the signaling traffic, consuming a fraction of MB per gossip cycle over a quite large network, where each link has a capacity by far larger than 1 Gb/s, has an impact completely negligible. Thus, the overhead of this background process is completely affordable by any modern broadband network, even for small values of T_{gossip}.

Now, we analyze the volume of traffic it consumes when the payload contains monitoring information with a size equal to $L = 1$ KB. The comparison is carried out between the OSP protocol (for the sizes of packets used by the protocol see [32]), the two configurations of the Leaf protocol (default and option 2) presented in Section III, with the same values of headers used before in this section, and a pub/sub distribution solution using Kafka (with an overhead of about

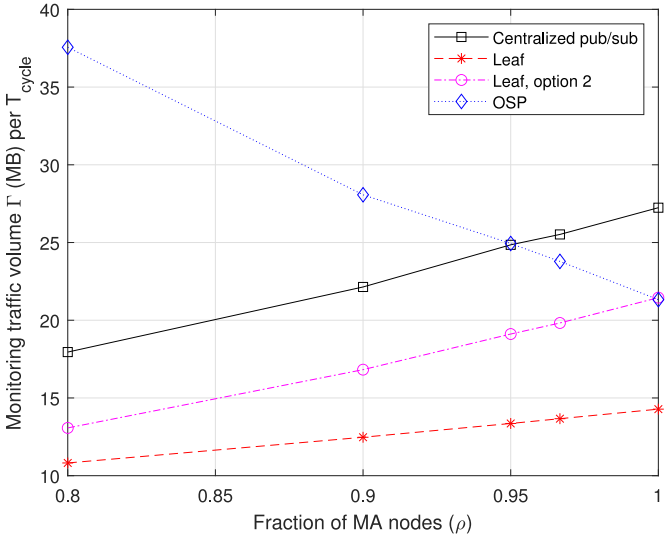


Fig. 8. Total amount of monitoring traffic produced during each monitoring period T_{cycle} by the four compared approaches (publish/subscribe with Kafka, Leaf, Leaf option 2, and OSP) as a function of the number of MAs in the overlay.

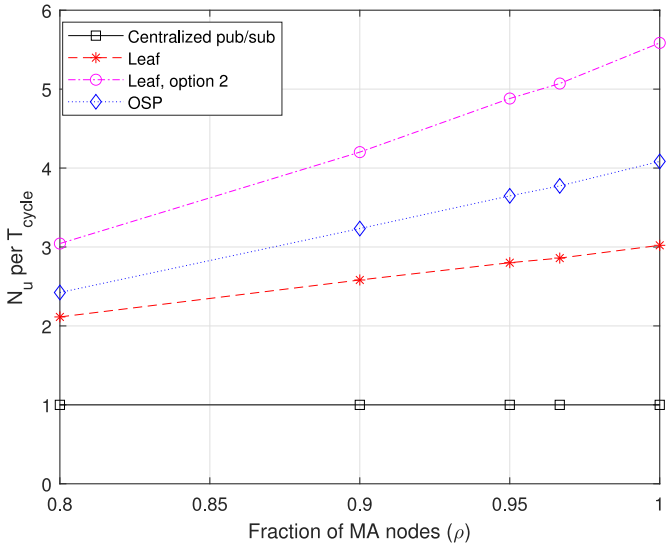


Fig. 9. Average number of updates received by each MA from other MAs in the overlay during each monitoring period T_{cycle} by the four compared approaches (publish/subscribe with Kafka, Leaf, Leaf option 2, and OSP) as a function of the number of MAs in the overlay.

1 KB including both headers and acknowledgements, evaluated by means of live captures) [22], [23]. While Leaf uses a single process to perform both network discovery via gossip and information distribution, in OSP two separated processes exists. One delivers information and the other is used for gossiping in background. For both protocols, each gossip exchange carries $H = 2$ PEs, since this value provides satisfying discovery times for all solutions with all configurations, as shown in Section IV-A. We focus on the total volume of traffic needed to distribute updates from all the MAs, which for Leaf is given by Γ_{Leaf} in (9), as a function of the number of involved MAs, ranging between the configuration of the *partial* topology $(\rho = 0.8)$ up to the *full* one $(\rho = 1)$.

Fig. 8 shows the number of Mbytes exchanged by MAs during a monitoring period to update the entries relevant to all others MAs, as a function of the fraction of MAs with respect to the nodes of the underlay. We define the monitoring period as the time needed to perform this process. For the Leaf approach, it is equal to T_{cycle}, whereas for the Leaf, option 2, it is equal to $\lceil T_{cycle}/2 \rceil$. As for the OSP, it uses a different mechanism to distribute such an information, not tied to a specific period, and the same for pub/sub approaches. It is clear that the Leaf approach is the solution requiring the lowest volume of traffic to update all the MAs in the network, estimated by Γ_{Leaf} in (9). It ranges between 50% and 60% of the traffic required by the centralized pub/sub solution, which can be considered the benchmark for the monitoring approach. As expected, all approaches increase the amount of traffic with the value of ρ but OSP. This is a specific feature of that protocol, which works better for topologies that are meshed as much as possible. In any case, the Leaf approach always provides higher efficiency than OSP, which is instead able to equate that of Leaf, option 2, only for $\rho = 1$. However, for $\rho = 0.8$, the required traffic is more than double that of the pub/sub solution. Thus, this approach is not suitable for generic topologies. As for the second option of the Leaf approach, it always requires a volume of traffic inferior than 80% of the pub/sub approach. In addition, it is able to complete the process in about half the time required by Leaf, which is valuable.

The reason of the lower efficiency of Leaf option 2 with respect to the default version of Leaf can be explained by looking at Fig. 9, which reports the average number of updates received by a MA during T_{cycle} as a function of ρ. Clearly, the information in the centralized pub/sub approach is updated exactly once per monitoring period, since each MA publish this information to the broker one time only, and all the other MAs subscribing the topic will receive it. Instead, in the other distributed approaches, this number is generally larger than 1. In particular, when using the Leaf approach, option 2, with its ability to intercept not only Registration messages, but also Responses, each MA may receive multiples updates from the same MA. This is true especially for core MAs, which may receive multiple updates from the same *initiator* when the *responders* are the relevant leaves. While this approach allows halving the monitoring period with respect to the base version of the Leaf, it implies larger overhead.

A possible solution, which we will explore as future work, is the possibility, for a given MA, to stop sending updates to the leaves from which it receives at least 2 updates during a monitoring period T_{cycle}. In fact, one of these updates are due to its gossiping to these leaves (acting as its *responders*), whereas the others will be triggered by intercepting Registration or Response messages as *forwarder*. By eliminating the direct messages, the overhead should decrease, without compromising the process of information distribution. The presence of a lifetime timer larger than T_{cycle} allows avoiding accidental cancellation of leaves. On the other hand, the possibility to receive more frequent updates enables a prompter distribution of state changes of the computing cluster monitored by MAs, and this holds for both versions of the Leaf approach.

V. Conclusion

In this paper we showed a proposal for providing a robust, distributed monitoring service for a 6G network architecture. The proposed solution, based on the concepts of gossip and network softwarization, does not depend on the number of cloud-native SF instances running in computing clusters, and it can adapt to changes in the (virtualized) network topology. Also, it fits the concept of service slice in modern network architectures very well, since each slice can build its virtual topology, including only a subset of monitoring agents. Thus, it can be used as a building block to realize scalable monitoring solutions in forthcoming 6G networks. Given the protocol properties, our solution can be adopted also by using virtual links interconnecting data center tenants offered by differed cloud providers, extending the overall scope even beyond the 6G network, guaranteeing a significant implementation flexibility. Finally, incremental deployment is possible, favoring its adoption in real settings.

We showed that the proposed solution, named Leaf, can nearly halve the volume of traffic exchanged to distribute state information with respect to state of the art solutions, based on centralized publish/subscribe solutions.

Future work will pursue the complete system implementation through open source software programs, as well as additional optimization for the Leaf option 2 approach.

References

[1] M. Chafii, L. Bariah, S. Muhaidat, and M. Debbah, "Twelve scientific challenges for 6G: Rethinking the foundations of communications theory." 2022. [Online]. Available: https://arxiv.org/abs/2207.01843.

[2] E. C. Strinati et al., "6G: The next frontier: From holographic Messaging to artificial intelligence using subterahertz and visible light communication," *IEEE Veh. Technol. Mag.*, vol. 14, no. 3, pp. 42–50, Sep. 2019.

[3] S. D. A. Shah, M. A. Gregory, and S. Li, "Cloud-native network slicing using software defined networking based multi-access edge computing: A survey," *IEEE Access*, vol. 9, pp. 10903–10924, 2021.

[4] Q. Li et al., "6G cloud-native system: Vision, challenges, architecture framework and enabling technologies," *IEEE Access*, vol. 10, pp. 96602–96625, 2022.

[5] K. B. Letaief, Y. Shi, J. Lu, and J. Lu, "Edge artificial intelligence for 6G: Vision, enabling technologies, and applications," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 1, pp. 5–36, Jan. 2022.

[6] T. Hewa, G. Gür, A. Kalla, M. Ylianttila, A. Bracken, and M. Liyanage, "The role of blockchain in 6G: Challenges, opportunities and research directions," in *Proc. 2nd 6G Wireless Summit (6G SUMMIT)*, 2020, pp. 1–5.

[7] M. Corici and T. Magedanz, *One Layer to Rule Them All–Data Layer-oriented 6G Networks*. Hoboken, NJ, USA: Wiley, 2021, ch. 13, pp. 221–233. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119765554.ch13

[8] T. Gonzalez, "An efficient algorithm for gossiping in the multicasting communication environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 14, no. 7, pp. 701–708, Jul. 2003.

[9] N. Janbi, I. Katib, A. Albeshri, and R. Mehmood, "Distributed artificial intelligence-as-a-service (DAIaaS) for smarter IoE and 6G environments," *Sensors*, vol. 20, no. 20, p. 5796, Oct. 2020. [Online]. Available: https://doi.org/10.3390/s20205796

[10] J. Tu, J. Zhou, and D. Ren, "An asynchronous distributed training algorithm based on gossip communication and stochastic gradient descent," *Comput. Commun.*, vol. 195, pp. 416–423, Nov. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366422003498

[11] G. Saldamli, C. Upadhyay, D. Jadhav, R. Shrishrimal, B. Patil, and L. Tawalbeh, "Improved gossip protocol for blockchain applications," *Clust. Comput.*, vol. 25, no. 3, pp. 1915–1926, Jan. 2022. [Online]. Available: https://doi.org/10.1007/s10586--021-03504-z

[12] "Network functions virtualisation (NFV); management and orchestration," ETSI, Sophia Antipolis, France, ETSI GS NFV-MAN 001 V1.1.1, Dec. 2014.

[13] S. Hashima et al., "On softwarization of intelligence in 6G networks for ultra-fast optimal policy selection: Challenges and opportunities," *IEEE Netw.*, early access, Feb. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9717267

[14] M. S. Aslanpour et al., "Serverless edge computing: Vision and challenges," in *Proc. Aust. Comput. Sci. Week Multiconf.*, New York, NY, USA, 2021, p. 10. [Online]. Available: https://doi.org/10.1145/3437378.3444367

[15] N. Slamnik-Krijestorac, G. M. Yilma, M. Liebsch, F. Z. Yousaf, and J. Marquez-Barja, "Collaborative orchestration of multi-domain edges from a connected, cooperative and automated mobility (CCAM) perspective," *IEEE Trans. Mobile Comput.*, vol. 22, no. 4, pp. 2001–2020, Apr. 2023.

[16] V. Ziegler, H. Viswanathan, H. Flinck, M. Hoffmann, V. Räisänen, and K. Hätönen, "6G architecture to connect the worlds," *IEEE Access*, vol. 8, pp. 173508–173520, 2020.

[17] K. Dolui and S. K. Datta, "Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing," in *Proc. Global IOT Summit*, Geneva, Switzerland, Jun. 2017, pp. 1–6. [Online]. Available: http://www.eurecom.fr/publication/5193

[18] E. Peltonen et al., "6G white paper on edge intelligence," Oulu, Finland, 6G Res. Vis., White Paper, 2020.

[19] L. Baresi, D. F. Mendonça, M. Garriga, S. Guinea, and G. Quattrocchi, "A unified model for the mobile-edge-cloud continuum," *ACM Trans. Internet Technol.*, vol. 19, no. 2, p. 29, Apr. 2019. [Online]. Available: https://doi.org/10.1145/3226644

[20] L. Bittencourt et al., "The Internet of Things, fog and cloud continuum: Integration and challenges," *Internet Things*, vols. 3–4, pp. 134–155, Oct. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660518300635

[21] D. Milojicic, "The edge-to-cloud continuum," *Computer*, vol. 53, no. 11, pp. 16–25, Nov. 2020.

[22] R. Perez, J. Garcia-Reinoso, A. Zabala, P. Serrano, and A. Banchs, "A monitoring framework for multi-site 5G platforms," in *Proc. Eur. Conf. Netw. Commun. (EuCNC)*, 2020, pp. 52–56.

[23] R. Perez, J. Garcia-Reinoso, A. Zabala, P. Serrano, and A. Banchs, "An experimental publish-subscribe monitoring assessment to beyond 5G networks," *EURASIP J. Wireless Commun. Netw.*, vol. 2021, no. 80, pp. 1–27, Apr. 2021. [Online]. Available: https://jwcn-eurasipjournals.springeropen.com/articles/10.1186/s13638-021-01962-y

[24] M. Femminella and G. Reali, "Gossip-based monitoring of virtualized resources in 5G networks," in *Proc. IEEE INFOCOM IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, 2019, pp. 378–384.

[25] M. Mekki, S. Arora, and A. Ksentini, "A scalable monitoring framework for network slicing in 5G and beyond mobile networks," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 1, pp. 413–423, Mar. 2022.

[26] J. Liu, S. Mou, A. Morse, B. D. O. Anderson, and C. Yu, "Deterministic gossiping," *Proc. IEEE*, vol. 99, no. 9, pp. 1505–1524, Sep. 2011.

[27] X. Gao, X. Zhang, D. Shi, and G. Chen, "An efficient heuristic gossiping mechanism in ad hoc routing," in *Proc. CHINACOM*, 2007, pp. 978–982.

[28] A. Bagchi, S. Hakimi, and E. F. Schmeichel, "Gossiping in a distributed network," *IEEE Trans. Comput.*, vol. 42, no. 2, pp. 253–256, Feb. 1993.

[29] B. Haeupler, G. Pandurangan, D. Peleg, R. Rajaraman, and Z. Sun, "Discovery through gossip," in *Proc. ACM SPAA*, 2012, pp. 140–149.

[30] M. Femminella, R. Francescangeli, G. Reali, and H. Schulzrinne, "Gossip-based signaling dissemination extension for next steps in signaling," in *Proc. IEEE NOMS*, 2012, pp. 1022–1028.

[31] A. Ganesh, A.-M. Kermarrec, and L. Massoulie, "Peer-to-peer membership management for gossip-based protocols," *IEEE Trans. Comput.*, vol. 52, no. 2, pp. 139–149, Feb. 2003.

[32] M. Femminella, G. Reali, and D. Valocchi, "A signaling protocol for service function localization," *IEEE Commun. Lett.*, vol. 20, no. 7, pp. 1325–1328, Jul. 2016.

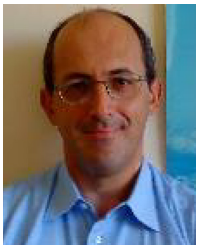
[33] R. Hancock, G. Karagiannis, J. Loughney, and S. V. den Bosch, "Next steps in signaling (NSIS): Framework," IETF, Fremont, CA, USA, RFC 4080, Jun. 2005.

[34] Y. Zhu, C. Hua, D. Zhong, and W. Xu, "Design of low-latency overlay protocol for blockchain delivery networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2022, pp. 1182–1187.

[35] L. Li, D. Huang, and C. Zhang, "An efficient DAG blockchain architecture for IoT," *IEEE Internet Things J.*, vol. 10, no. 2, pp. 1286–1296, Jan. 2023.

[36] J. Haddock, B. Jarman, and C. Yap, "Paving the way for consensus: Convergence of block gossip algorithms," *IEEE Trans. Inf. Theory*, vol. 68, no. 11, pp. 7515–7527, Nov. 2022.
[37] N. E. Manitara, A. I. Rikos, and C. N. Hadjicostis, "Privacy-preserving distributed average consensus in finite time using random gossip," in *Proc. Eur. Control Conf. (ECC)*, 2022, pp. 1282–1287.
[38] M. Kenyeres and J. Kenyeres, "Comparative study of distributed consensus gossip algorithms for network size estimation in multi-agent systems," *Future Internet*, vol. 13, no. 5, p. 134, May 2021. [Online]. Available: https://doi.org/10.3390/fi13050134
[39] T. Yu, L. Yu, and J. Xiong, "Observer-based distributed control of large-scale systems under gossip communication protocol," *Asian J. Control*, vol. 24, no. 2, pp. 956–972, Oct. 2021. [Online]. Available: https://doi.org/10.1002/asjc.2502
[40] Y. Gou, R. Wang, Z. Li, M. A. Imran, and L. Zhang, "Clustered hierarchical distributed federated learning," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2022, pp. 177–182.
[41] J. Zhou, J. Tu, and D. Ren, "An asynchronous distributed training algorithm based on gossip," in *Proc. 7th Int. Conf. Big Data Anal. (ICBDA)*, 2022, pp. 214–217.
[42] S. Voulgaris, M. Jelasity, and M. van Steen, "A robust and scalable peer-to-peer gossiping protocol," in *Proc. AP2PC*, 2005, pp. 47–58.
[43] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms*, 2nd ed. New York, NY, USA: McGraw-Hill, 2001.
[44] M. Femminella, G. Reali, and D. Valocchi, "Genome centric networking: A network function virtualization solution for genomic applications," in *Proc. IEEE Conf. Netw. Softw. (NetSoft)*, 2017, pp. 1–9.
[45] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. van Steen, "Gossip-based peer sampling," *ACM Trans. Comput. Syst.*, vol. 25, no. 3, p. 8, Aug. 2007. [Online]. Available: https://doi.org/10.1145/1275517.1275520

Mauro Femminella (Member, IEEE) received the master's and Ph.D. degrees in electronic engineering from the University of Perugia in 1999 and 2003, respectively. Since July 2022, he has been an Associate Professor with the Department of Engineering, University of Perugia. He is also the Representative of the University of Perugia in consortium CNIT, as well as the CNIT representative in 5G-PPP Working Groups, such as 5G CAM and Trials. He is the coauthor of more than 100 papers in international journals and refereed international conferences. His current research interests focus on molecular communications, big data systems, and architectures and protocols for 5G networks.



Gianluca Reali received the Ph.D. degree in telecommunications from the University of Perugia, Italy, in 1997, where he has been an Associate Professor with the Department of Engineering since January 2005. From 1997 to 2004, he was a Researcher with the Department of Electronic and Information Engineering, University of Perugia. In 1999, he visited the Computer Science Department, University of California at Los Angeles. His research activities include resource allocation over packet networks, wireless networking, network management, multimedia services, big data management, and nanoscale communications.

Open Access funding provided by 'Università degli Studi di Perugia' within the CRUI CARE Agreement