Off-path Signaling Extension for General Internet Signaling Transport Protocol

Mauro Femminella, Member, IEEE, Gianluca Reali, Member, IEEE,
Dario Valocchi, Roberto Francescangeli, and Henning Schulzrinne, Fellow, IEEE

Abstract—In this paper, we propose an off-path extension of the IETF Next Steps in Signaling (NSIS) protocol suite. Our proposal updates the NSIS transport layer. This way, the design of an NSIS-compliant application can leverage it without having to deal with low level signaling transport issues. In particular, we propose an extension of the General Internet Signaling Transport Protocol (GIST), including path decoupled signaling capabilities. This extension, executable with none or minimal IP node configuration, requires the definition of a GIST peer discovery protocol for making each GIST node aware of its GIST peers. This is indeed the first contribution of this paper. The second one consists of the delivery strategies of signaling messages, depending on the domain topological features. These strategies make use of the information collected during the GIST peer discovery phase and exploit the signaling interception capabilities of GIST. We have analyzed these novel signaling capabilities experimentally, after having implemented our solutions on Linux nodes. An upper bound to the signaling load is also determined. The conclusion is that off-path signaling features are made available by introducing a very small signaling load.

Index Terms—NSIS, signaling, off-path, network discovery, gossip, packet interception

I. INTRODUCTION AND MOTIVATIONS

The need of generalized off-path signaling mechanisms to support advanced service architectures has emerged since many years [2]. They allow simplifying the use of entities that are not in the path followed by data packets (datapath). Although this utility already emerged for supporting IP quality of service (QoS) and mobility (i.e. signaling with bandwidth broker and next access router, respectively), recently an increasing number of applications, that would benefit from the introduction of off-path signaling capabilities, have appeared. They may need different signaling distribution services:

• Signaling distributed around the signaling sender, referred to as bubble. It can be used by different applications, such as network management ones for sending messages to all neighbors within a given IP radius [3], [4], or virtual machine (VM) management applications [5] for collecting the status of neighboring data centers to migrate a VM.

• Signaling distributed around the signaling receiver, referred to as balloon. A typical service in this category is the massive data distribution through content delivery networks (CDNs), involving the home user equipment [6]–[8]. In fact, the parameter that mostly influences a CDN capacity is the number of peering points [9]. In order to make use of a highly distributed service architecture, it is necessary to exchange signaling messages with peers located a few IP hops away from the receiver to find suitable candidates at the edge of the network.

• Signalig distributed to network nodes a few IP hops away from the IP datapath, referred to as hose. This type of signaling is useful for networks made of programmable routers, which route packets towards processing nodes close to the datapath (see e.g. [10], [11]), even using the redirection facilities of Openflow [12], and for novel network architectures, which locate and use off-path entities such as proxies, as in [13]–[14].

In synthesis, the number of applications that could benefit from off-path signaling capabilities is large and rapidly increasing. Since their scope might include networks of different size and shape, it is preferable to introduce a common signaling suite, modular and easily adaptable to the topology of different domains.

The IETF Next Steps in Signaling (NSIS) [15] Working Group was formed in 2001. It had the goal of designing a new generic Internet signaling protocol suite, in order to overcome the shortcomings of other signaling protocols, such as SIP and SS7, the design of which was targeted to specific needs, and may not be easily adaptable to the needs of new applications.

The NSIS framework was designed to be modular, easily extendable, secure, and customizable. However, since NSIS was primarily designed to solve RSVP issues, its QoS-aware nodes discovery process is tied to the datapath. In fact, it can be used to install, modify, and remove states both on the two communicating hosts (end-to-end signaling), and on intermediate nodes (path-coupled signaling). This is typically achieved by exchanging end-to-end signaling messages, having an IP Router Alert Option (RAO) [16] in their header, intercepted by NSIS-capable nodes along their path to the destination. However, the NSIS architecture is flexible [17], and signaling message routing is controlled by the so-called Message Routing Method (MRM). An MRM is the algorithm used by NSIS to both discover peers and route signaling.
messages. Thus, in order to decouple signaling messages from datapath, it is necessary to create new MRMs. In this way, new signaling applications can be made available to NSIS nodes.

We propose a new off-path signaling MRM for the General Internet Signaling Transport (GIST) protocol \[18\], which is the IETF-defined version of the NSIS Transport Layer Protocol. This solution retains all the benefits introduced with the NSIS signaling architecture, and provides off-path signaling capabilities within scopes specified by any suitable metrics, such as the number of IP hops. The proposed scheme makes use of the packet interception capabilities of GIST.

Since GIST nodes may be incrementally deployed in autonomous systems (ASs), they tend to be scattered, forming an overlay. Each GIST node needs to know the GIST overlay where its signaling messages must be distributed. To solve this issue, we propose also a gossip-based GIST node discovery protocol, which (once again) makes use of the packet interception capabilities of GIST in order to minimize network overhead and discovery time. This protocol, which is the second contribution of this paper, is used to dynamically and asynchronously bootstrap any GIST network by collecting node capabilities and calculating different metrics.

In order to evaluate the effectiveness of both gossip-based discovery and off-path signaling delivery, we used a network of 60 NSIS-enabled Linux nodes. The proposed GIST extensions have been implemented within NSIS-ka \[19\], an open source NSIS implementation provided by the Karlsruhe Institute of Technology. The performance of the gossip protocol on larger topologies have been evaluated by using a custom simulator.

The paper is organized as follows. Section II describes the NSIS architecture, by focusing on GIST. Section III illustrates our gossip-based NSIS node discovery process and the off-path signaling extensions. Section IV shows the experimental testbed and results. Section V discusses related work and alternative solutions. Finally, section VI draws conclusions.

II. BACKGROUND: THE NSIS PROTOCOL FAMILY

The NSIS protocol suite divides the signaling functions into two layers. The upper layer, called NSIS Signaling Layer Protocol (NSLP), implements the application signaling logic. The lower layer, called NSIS Transport Layer Protocol (NTLP), is in charge of delivering the NSLP protocol messages to the next NSIS node on path. It is thus necessary to discover both the next-hop NSIS node, which may differ from the next IP node, and the transport and security services available for fulfilling the signaling application requirements. GIST is the IETF-defined version of the NTLP. GIST transport services make use of existing protocols in the TCP/IP suite. Normally UDP is used for delivering unsecure signaling, TCP is needed when reliability is required, and TLS over TCP solution provides secure and reliable signaling transport. GIST only delivers NSLP messages hop-by-hop between pairs of neighboring NTLP signaling nodes, whereas the end-to-end signaling functions, if needed, are provided by NSLP.

Before starting a signaling session, GIST peers have to create a Message Association (MA) by using the information transported in the so-called Network Layer Information (NLI) object of GIST packets, such as the unique identifier of the GIST node (Peer Identity, PI) and one of its IP addresses.

Although only two NSLP protocols are currently standardized by IETF RFCs (Quality-of-Service signaling \[20\] and NAT/firewall traversal \[21\]), many others have been implemented, such as the NettServ \[22\] NSLP.

NSIS has been primarily designed for managing states on nodes lying on data paths. For this purpose, NTLP messages may be intercepted at NSIS-capable nodes on path. In particular, the GIST protocol allows specifying messages routing policies through MRMs. Two MRMs are currently specified:

- Path-Coupled MRM, which routes signaling messages through the data path;
- Loose End MRM, used for preconditioning firewalls and NAT states when data destinations lie behind them.

GIST messages include a Message Routing Information (MRI) object, that allows NSIS nodes to identify the MRM to be used. For example, in case of a Path-Coupled MRM, GIST packets are intercepted by NSIS nodes on-path and then re-sent towards the destination, after being processed by the NSLP entities.

Three encapsulations have been defined for GIST:

- D-mode: UDP transport without packet interception;
- C-mode: TCP transport without packet interception;
- Q-mode: UDP transport with port 270 assigned by IANA (IP RAO optional \[16\]), and packet interception at GIST nodes. Legacy IP nodes transparently forward packets towards their destination.

In order to limit the complexity of the signaling architecture, the NSIS WG imposed two restrictions:

- initial focus only on MRMs for path-coupled signaling;
- no multicast support.

These restrictions can be lifted by taking advantage of both the modular architecture of NSIS and GIST, and the possibility of defining new MRMs.

III. OFF-PATH GIST PROTOCOL EXTENSION

In this section, we first present the algorithms used to implement the gossip-based GIST discovery process. Then, we illustrate the proposed off-path MRM extension of GIST, by focusing on the algorithms that make use of the information collected by the discovery process in order to distribute signaling over the domain topologies defined in Section II.

A. Gossip Discovery Process

The approach used to discover GIST nodes consists of a gossip protocol extension to GIST, which leverages the GIST packet interception. Gossip sessions are round-based and asynchronous. They are established between two nodes, an initiator and a responder, through a three-way handshake, which makes use of three new GIST 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. As in other gossip protocols (e.g., see \[23\]), both
the Registration and the Response messages include a list of GIST peers that the initiator and the responder may want to share with each other, referred to as peers to share list (PTS). Therefore, each node can establish gossip sessions with other (possibly unknown) nodes on subsequent rounds. Fig. 1 shows the content of messages exchanged during a gossip session.

Differently from other gossip protocols (see section V for details), the GIST packet interception capabilities allows the Registration message, encapsulated in Q-mode, to be received and processed not only by the responder, but also by the GIST nodes on its path (see Fig. 2). This way, these nodes can participate to the discovery process by sending Response messages towards the initiator, sharing their own PTS list.

We envision and analyze two managing policies for intercepted Registration messages in intermediate GIST nodes. The first one, referred to as Q-forward, consists of forwarding the message towards the responder after having sent a Response back to the initiator (Fig. 2). By handshaking with each node, the initiator can evaluate its downstream distance from all nodes along the path, in terms of both GIST/IP hops and, roughly, latency for reaching each of them. Since IP routing could be asymmetric, these distances may not be valid in the reverse direction. Thus, the only reliable upstream information, collected by the responder and intermediate GIST nodes, is the initiator identity.

The second policy, denoted Q-drop, allows only the first GIST node on the path to intercept the Registration and return the Response. Then, the Registration is discarded and not forwarded downstream, as illustrated in Fig. 2.2.

The design goals of these two approaches are different. The Q-forward policy aims to discover overlay paths and evaluate the associated metrics. This information can be used by the off-path signaling distribution algorithms presented in section III-B. The goal of the Q-drop policy is simply to allow the initiator to collect the identities of the GIST neighbors, that is GIST nodes at one GIST hop away, together with the number of IP hops within this GIST hop, and a rough estimation of the communication latency. Although the Q-drop policy allows achieving a partial knowledge of the network, it scales well in networks with many nodes. In addition, it provides enough information to run the flooding-based off-path distribution strategy illustrated in section III-B. We now illustrate the (common) procedure used to bootstrap the GIST protocol when a node is turned on. It is inspired by [24]. Then, we detail two gossip solutions, called Leaf and One-hop, based on Q-forward and Q-drop policies, respectively.

1) Gossip bootstrap procedure for GIST nodes: When a GIST node is turned on, its list of reachable GIST nodes is empty. Thus, it is necessary to ask another always-on node, called tracker, an initial list of active GIST nodes to gossip with. Thus, the tracker acts as the first GIST responder (node N4 in Fig. 2). A tracker address must therefore be statically configured in all GIST nodes. A different solution consists of making use of an IP-to-ASN public mapping service. Any turned-on GIST node can use its own IP address to retrieve the Autonomous System Number (ASN), referred to as asn1, from this mapping service. Then, the node can obtain the tracker IP address by issuing a DNS query to resolve asn1.nsis.org. This procedure needs the creation of a generic container domain, such as “nsis.org” to which AS providers can register the IP addresses of the trackers of their networks, similarly to [25].

After this initial procedure, a GIST node knows (at least) one additional GIST node, and can periodically establish a gossip session with it, in order to update its PTS list. Clearly, a GIST node cannot know when it has discovered all the other GIST nodes in the network (end of discovery phase). It assumes that when it does not learn the identity of any new GIST node for a given number of subsequent cycles, it enters the steady phase. In this case, it can increase the gossip session period, in order to limit the consumption of network resources.
2) Leaf-based gossip protocol: Before entering the protocol details, we present a mathematical model of the GIST gossip protocol based on the Q-forward policy. Our network model consists of a graph of GIST nodes only, referred to as GIST overlay, denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. $\mathcal{V}$ is the set of GIST nodes with cardinality $K = |\mathcal{V}|$, and $\mathcal{E}$ is the set of the undirected edges. The IP routing of GIST packets, which determines the elements in $\mathcal{E}$ connecting GIST peers, makes use of the underlying IP routing protocols, which we assume to be based on shortest-path algorithms. GIST can intercept packets when the Q-mode encapsulation is used, and applies the Q-forward policy to these packets. We define a path $\pi_{ij} = \{i, k, ... j\}$ as the ordered sequence of GIST nodes on the IP 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 GIST nodes visited by a packet sent by the peer $i$ towards the peer $j$. We define $\mathcal{S} = \{s_{ij}|i, j \in \mathcal{V}\}$. For each GIST node $i \in \mathcal{V}$, we also define the set of its GIST neighbors as $\mathcal{N}_i = \{j \in \mathcal{V}|s_{ij} = \{j\}\}$, with $A_i = |\mathcal{N}_i|$. Let us focus on the discovery phase. It allows all GIST nodes in $\mathcal{V}$ to receive the identities of the other GIST nodes and to evaluate the relevant metrics. The minimization of the discovery time translates in minimizing the number of gossip rounds necessary to the node $i$ to contact all the other GIST nodes in $\mathcal{V}$ by leveraging the interception capability of GIST. In addition, for each GIST node $i \in \mathcal{V}$ we define the single-source shortest-path tree $\mathcal{T}_i$ rooted at $i$, $\mathcal{T}_i$ identifies the GIST nodes on the (shortest) IP path towards any other node $k \in \mathcal{V}$. An example of $\mathcal{T}_i$ is drawn in bold in Fig. 3, where $i = 1$. We say that a node $h \in \mathcal{V}$ is a leaf for the tree $\mathcal{T}_i$, that is $h \in s_{ij} \iff h = j$. We denote as $\mathcal{L}_i$ the set of leaf nodes for $i$, and $M_i = |\mathcal{L}_i|$. Paths associated to leaves for node 1 are shown by red dashed arrows in Fig. 3. 

Our proposed solution of the problem $\mathcal{C}_1$ 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 GIST nodes in $\mathcal{G}$, together with the relevant metrics, thanks to the Q-mode encapsulation and Q-forward policy. Thus, our solution is aimed to quickly discover all Leaf Peers (LPs) of the tree associated with each node in $\mathcal{V}$. A formal proof of this statement is provided by Theorem 1. 

Theorem 1: The optimal solution $\mathcal{D}_i^*$ to the set cover problem $\mathcal{C}_1$ 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, $\exists y \in \mathcal{L}_i | z \in s_{iy}$. Thus, from the shortest path routing assumption it follows that $s_{iz} \subseteq s_{iy}$. Since $y \in \mathcal{L}_i \subseteq \mathcal{D}_i^*$, $\mathcal{D}_i^* - \{z\}$ is still a solution of $\mathcal{C}_1$ (see (2)) but with a lower cost than $\mathcal{D}_i^*$ (see (1)). Consequently $\mathcal{D}_i^*$ cannot be an optimal solution for $i$. $\square$ 

Similarly, it is very easy to show that $\mathcal{L}_i$ is the solution also for the optimization problem modeling the network resource consumption during the discovery phase, denoted $\mathcal{C}_2$:

$$\min \sum_{i=1}^{K} \sum_{j \in \mathcal{B}_i} p_{ij}q + \sum_{z \in s_{ij}} (p_{zz}r + p_{z,a})$$

subject to

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

The solution of this problem, that is the identification of the minimum sets $\mathcal{D}_i, i \in \mathcal{V}$, provides a solution of the GIST discovery problem for all GIST peers. In fact, $|\mathcal{D}_i|$ is the minimum number of gossip rounds necessary to the node $i$ to contact all the other GIST nodes in $\mathcal{V}$ by leveraging the interception capability of GIST. In addition, for each GIST node $i \in \mathcal{V}$ we define the single-source shortest-path tree $\mathcal{T}_i$, rooted at $i$. $\mathcal{T}_i$ identifies the GIST nodes on the (shortest) IP path towards any other node $k \in \mathcal{V}$. An example of $\mathcal{T}_i$ is drawn in bold in Fig. 3, where $i = 1$. We say that a node $h \in \mathcal{V}$ is a leaf for the tree $\mathcal{T}_i$, that is $h \in s_{ij} \iff h = j$. We denote as $\mathcal{L}_i$ the set of leaf nodes for $i$, and $M_i = |\mathcal{L}_i|$. Paths associated to leaves for node 1 are shown by red dashed arrows in Fig. 3. 

Our proposed solution of the problem $\mathcal{C}_1$ 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 GIST nodes in $\mathcal{G}$, together with the relevant metrics, thanks to the Q-mode encapsulation and Q-forward policy. Thus, our solution is aimed to quickly discover all Leaf Peers (LPs) of the tree associated with each node in $\mathcal{V}$. A formal proof of this statement is provided by Theorem 1.
to optimize by contacting only specific nodes (often still unknown) to limit the number of necessary gossip rounds. For LPs, as in Fig. [2],. For this reason, we call this solution Leaf-based. In addition, each node, in order to let other nodes quickly discover their LPs, exchanges only the identities of potential leaves in the PTS field of Registration and Response messages (Fig. [1] “Node-List”). To this aim, we have defined simple, lightweight, and soft-state structures storing GIST peer information at each GIST node. The former, called peer table (PeT), stores the identities of the other GIST peers, together with their associated metrics (peer element, PE). The latter, called path table (PaT), stores in node i the ordered sequence of PEs in sij, as new GIST nodes are discovered and contacted. 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.

It is worth to note that a given node z, which is a leaf for i, not necessarily is a leaf also for j, which receives the PTS list sent by i. Finally, it may happen, especially in the initial gossip rounds, that a newly activated GIST node knows just a limited number of peers, thus the identities it shares could not be true leaves.

We now detail the algorithms executed in GIST nodes.

The main loop executed by a GIST initiator is shown in Fig. [3] which includes the management procedure of the gossip timer \( T_{\text{gossip}} \), briefly sketched in section III-A1. \( T_{\text{gossip}} \) is increased if no PeT changes are done for a number \( \text{maxCounter} \) of subsequent gossip sessions (see lines 8-16), and restored to its initial value when a change happens (lines 5 and 6). When the timer expires, this procedure calls the function \( \text{gossipSession} \), which is the core procedure executed by an initiator, illustrated in Fig. [5] and detailed below.

The initiator must select two types of peers stored in the PeT: the so-called peer to gossip (PTG), which is the responder, selected by invoking the function \( \text{getPeerToGossip} \) (lines 1-7), and the PTS list, which includes the identities of the PEs to share with the PTG and any intercepting nodes (lines 8-10), through the invocation of the function \( \text{getPeers} \). Since gossip sessions must be established only with leaves, the algorithms implemented to select PTG and PTSs aim at this goal. In the PE entry of the PTG stored in the initiator, the flag \( \text{isTried} \), used for managing soft states, is set to “true”. The gossip Responses received by the initiator are managed by the loop described in lines 13-36. When the initiator receives the PTS list within a Response sent by a remote peer, it performs two actions:

- each element of the received PTS not already present in the PeT is added, and its flags \(<\text{isGossiped, isContacted}>\) are 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 PTS list is a good candidate for being selected as future PTG;
- each GIST intercepting peer on the path to the PTG is added to the PeT, if not present, together with its metrics, and the relevant flag \( \text{isContacted} \) is set to “true”. This peer is not a good candidate for future selection of the PTG or a PTS element, since it is not an LP for the initiator.

**Procedure mainLoop**

**Input:** \( \text{minT}_{\text{gossip}}, \text{maxT}_{\text{gossip}}, \text{maxCounter} \)

1. \( T_{\text{gossip}} \leftarrow \text{minT}_{\text{gossip}} \)
2. \( \text{gossipSessionCounter} \leftarrow 1 \)
3. **loop**
4. if new peer arrived during past gossip session \( \land \) PeT contains uncontacted peers then
5. \( T_{\text{gossip}} \leftarrow \text{minT}_{\text{gossip}} \)
6. \( \text{gossipSessionCounter} \leftarrow 1 \)
7. **else**
8. if \( \text{gossipSessionCounter} \leq \text{maxCounter} \) then
9. \( \text{gossipSessionCounter}++ \)
10. **else**
11. if \( 2T_{\text{gossip}} \leq \text{maxT}_{\text{gossip}} \) then
12. \( T_{\text{gossip}} \leftarrow 2T_{\text{gossip}} \)
13. **else**
14. \( T_{\text{gossip}} \leftarrow \text{maxT}_{\text{gossip}} \)
15. **end if**
16. **end if**
17. **end if**
18. start \( T_{\text{gossip}} \) timer
19. call Procedure \( \text{gossipSession} \)
20. if \( T_{\text{gossip}} \) timer is not expired then
21. wait \( T_{\text{gossip}} \) timer expiration
22. **end if**
23. **end loop**

Fig. 4. Main loop of a GIST gossip initiator.

Two issues are still open. The first is the metric evaluation and path construction in the PaT maintained by the initiator. The second is the selection of PTG and PTS elements.

For what concern the first issue, the Registration message includes an NLI element (Fig. [1]), used to compute the distance between the initiator and the PTG in terms of IP and GIST hops [18]. The initiator initializes a \( \text{TTL} \) value in the IP hop field of the NLI (\( \text{TTL}_{\text{NLI}} \)) of the Registration message, which cannot be modified by intercepting nodes. This value is copied in the TTL field of the IP header (\( \text{TTL}_{\text{IP}} \)), which is decremented at each IP hop. Finally, the \( \text{TTL} \) value is also copied in the GIST hop field of the GIST header (\( \text{HOP}_{\text{GIST}} \)), which is decremented at each GIST hop.

Once the Registration packet is intercepted by a GIST node, either a forwarder or a responder, it computes its IP and GIST distances from the initiator. The IP distance is evaluated as \( \text{TTL}_{\text{NLI}} - \text{TTL}_{\text{IP}} \) and written in the NLI of the Response, whereas the GIST distance is \( \text{TTL}_{\text{NLI}} - \text{HOP}_{\text{GIST}} \) and written in the GIST hop field (lines 2-8 of pseudo-code shown in Fig. [6]). The Response message is sent by using the D-mode encapsulation, so as to not be intercepted and, consequently, to keep the GIST hop field untouched.

In turn, the initiator updates a temporary path list (\( \text{peerList} \)) as it receives Responses. The position in the \( \text{peerList} \) of a peer is exactly its distance in GIST hops. The procedure is completed when the GIST hop distance of the PTG is equal to the number of received responses (size of the \( \text{peerList} \)), and the last element of that list is exactly the PTG (note that
Procedure gossipSession
Input: (peerList, pathList)
Output: pathList
1: if this is the first gossip session then
2: peerToGossip ← tracker
3: else
4: call procedure getPeerToGossip
5: Input: peerTable
6: Output: peerToGossip
7: end if
8: call procedure getPeers
9: Input: Number of peers to share, peerToGossip
10: Output: peersToShare: list of peers to share
11: Send peerToShare list to peerToGossip
12: peerToGossip.isTried ← true
13: peerList: empty list of element
14: while $T_{gossip}$ is not expired do
15: Receive sharedPeers from peer in a Response
16: for all element ∈ sharedPeers do
17: if element $\notin$ peerTable then
18: add element to peerTable
19: element.isGossiped ← true
20: element.isContacted ← false
21: end if
22: end for
23: if peer $\notin$ peerTable then
24: add peer to peerTable
25: peer.isContacted ← true
26: else
27: update peer metrics
28: end if
29: append peer to peerList
30: order peerList basing on GIST Hop
31: if last (peerList) == peerToGossip ∧
   |peerList| == peerToGossip.gistHop then
32: send gossip ack to peer
33: break while
34: else
35: send gossip ack to peer
36: end if
37: end while
38: call procedure updatePathList

Fig. 5. Gossip session initiator procedure.

Procedure gossipDaemonLoop
1: loop
2: Receive a gossip Registration message
3: $TTL_{IP}$ ← packet IP header TTL field
4: $HOP_{GIST}$ ← message.GISThopCount
5: $TTL_{NLI}$ ← message.nli.ipttl
6: create a gossip Response message gossipResponse
7: $gossipResponse.nli.ipttl$ ← $TTL_{NLI} − TTL_{IP}$
8: $gossipResponse.GISThopCount$ ← $TTL_{NLI} − HOP_{GIST}$
9: call procedure getPeers
10: Input: (Number of peers to share, gossip initiator)
11: Output: $PTS$: list of peers to share
12: $gossipResponse.peersToShare$ ← $PTS$
13: send gossipResponse to the gossip initiator
14: if message.destination is not ∈ myAddresses then
15: forward message toward destination
16: end if
17: for all element ∈ message.sharedPeers do
18: if element $\notin$ peerTable then
19: add element to peerTable
20: element.isGossiped ← true
21: element.isContacted ← false
22: end if
23: end for
24: if initiator $\notin$ peerTable then
25: add initiator to peerTable
26: initiator.isGossiped ← false
27: initiator.isContacted ← false
28: end if
29: loop
30: Receive a gossip acknowledgement
31: break inner loop
32: end loop
33: end loop

Fig. 6. Daemon loop executed from the GIST forwarder and the GIST responder.

The selection of PTS elements is a common process to initiator, forwards, and responder. Assume that the maximum size of the PTS list is $H$, which is a protocol design parameter. Since the Leaf-based gossip protocol aims at 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 PTS list 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 contacted the selecting node, 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 to making use of peers already contacted, which are not LPs, given that all nodes should preferably share LPs.

PTG is selected randomly from three priority lists. The first list, referred to as high priority, includes uncontacted PEs with the flag $isGossiped$ set true, since they are most likely LPs. The second list, referred to as low priority, includes uncontacted PEs with the flag $isGossiped$ set false, that are not likely to be LP. 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 by a single Registration message.

Since PE states are soft, they are removed if not refreshed. The lifetime value depends on the number of paths in the PaT (pathList), and is set equal to maxT_{gossip} × max|pathList|, 1 × (1 + Δ), where Δ is a parameter used to avoid accidental PE cancellations. The PaT consistency is guaranteed by updates done when a new path is collected during a gossip session. For space limitations, we do not report details of the function updatePathList, which merges, updates, or truncates paths already present in the pathList as a consequence of a gossip session (line 38 of Fig. 5).

3) One-hop gossip protocol: An alternative network discovery approach is to limit the scope of the gossip exchanges to the neighboring GIST peers only, in order to limit the communication overhead. This approach will be referred to as One-hop. In the One-hop strategy, Registration messages are encapsulated in Q-mode and managed by the Q-drop policy by GIST forwarders (see Fig. 2.2). We can again model the network discovery problem as a set covering problem (C3):

\[
\min \sum_{i=1}^{K} |\mathcal{P}_i|, \\
\text{subject to} \\
\mathcal{N}_i \subseteq \bigcup_{j \in \mathcal{P}_i} \mathcal{S}_{ij}, \forall i \in \mathcal{V},
\]

where \( \mathcal{P}_i \subseteq \mathcal{U}_i \) is the set of destinations of the Registration messages sent by node \( i \), \( \mathcal{N}_i \) is clearly an optimal solution of (C3), since the inclusion in (6) becomes an equality. However, any set of destinations \( \mathcal{F}_i \neq \mathcal{N}_i, \mathcal{F}_i \subseteq \mathcal{U}_i \) respecting the condition (6) with the additional constraint

\[
s_{ij} \cap s_{ik} = \emptyset, \forall j, k \in \mathcal{F}_i,
\]

is also an optimal solution of (C3), without any additional network overhead with respect to the trivial solution \( \mathcal{N}_i \) due to the usage of the forwarding policy Q-drop. In fact, while the condition (6) ensures that the union of sequences \( s_{ij} \) with destinations \( j \in \mathcal{F}_i \) is a cover of the set \( \mathcal{N}_i \), the condition (7) ensures that each GIST neighbor is counted just once, and thus the result of the sum (5) is \( \sum_{i=1}^{K} \mathcal{A}_i \). Nevertheless, even if the problem can be easily solved by an “oracle” having a complete knowledge of the overlay \( \mathcal{G} \), it is all but trivial to solve in operation, for the same reasons related to the access to routing information mentioned in section III-A2.

In order to efficiently discover destinations while respecting both (6) and (7), each peer classifies PEs as follows:

- **Neighbor peer**: a peer at 1 GIST hop distance, with metric values obtained by a complete gossip session. Its flags \(<\text{isContacted}, \text{isTried}>\) are set to \(<\text{true}, \text{true}>\);
- **Unreachable peer**: a peer located at a GIST hop distance greater than 1, that the peer unsuccessfully tried to contact in a gossip session. Metric values are thus undefined. Its flags \(<\text{isContacted}, \text{isTried}>\) are set to \(<\text{false}, \text{true}>\);
- **Uncontacted peer**: a peer with no contact attempt. Its flags are set to <false, false>.

A node tries to contact still uncontacted peers first (PTG in discovery phase), and then all other ones. The use of distinct flags for neighboring and unreachable peers allows contacting only neighbors in the steady phase, but not unreachable peers that are behind already discovered neighbors. The transition from the discovery to the steady phase is managed as in the Leaf-based solution, according to the algorithm illustrated in Fig. 4. The PTS selection is random, since there is no value in limiting the choice to one or two of the above categories. In fact, differently from leaves, the sets of GIST neighbors could be quite different for two neighboring peers exchanging their PTS list each other.

Although the number of peers to contact is significantly lower than in the leaf-based approach (in general \( A_i \ll M_i \)), the fact that at each gossip cycle it is possible to contact only one peer (i.e., the first on the route to the destination) could be disadvantageous, since it allows to classify just one peer as unreachable/neighbor at a time.

An important feature of the One-hop discovery process is the management of the unreachable peer identities. In order to avoid storing useless PEs with no valid metrics associated, we limit the maximum number of unreachable peers in the PeT. The unreachable peers in the PeT are managed according to a Least Recently Used (LRU) strategy. Without this limitation, the size of the PeT could increase up to the number of GIST nodes in the network. However, storing a small number of unreachable peer identities is useful. In fact, it allows avoiding to start gossip handshakes towards unreachable peers already checked, which would only increase the network overhead and, eventually, the discovery time of neighboring peers. In addition, having unreachable peers to check would not allow the algorithm to switch from the discovery to the steady phase (i.e., to relax the \( T_{gossip} \) timer, lines 4-6 in Fig. 4), even if all neighbors for that node have been discovered.

Another important feature of the One-hop discovery process is the management of multiple IP addresses of GIST peers, as it happens for routers or multihomed servers. When the NLI of a received gossip message includes the PI of a peer already present in the PeT and labeled as unreachable, if the IP address in the NLI is different from the one stored in the PeT, this address updates the stored one, and the PE is set as uncontacted. Consequently, a new attempt to contact the PE may be done by using the newly inserted IP address in subsequent gossip sessions. This feature is necessary since IP nodes with multiples interface in densely connected networks can be reached through different paths depending of the used IP address, when the path cost is equal.

PE are modeled through soft states also in this solution.

B. Off-path GIST dissemination

We propose a new MRM that allows sending signaling messages decoupled from the data path. Our objective is to distribute GIST signaling messages within off-path distribution domains, including peers which meet specific conditions. Clearly, distribution domains considered in what follow are related to data paths, but not coincide with them.
We will make use of the following notation, where $r$ is a radius expressed in IP hops:

- $d_i(r) = \{j \in V : p_{ij} = r\}, i \in V$;
- $c_i(r) = \{j \in V : p_{ij} \leq r\} = \bigcup_{l=1}^{r} d_i(l)$, with $i \in V$.

The proposed off-path dissemination types are as follows:

- **Bubble**: a region around an initiator $i$. Signaling messages are delivered to $c_i(r)$;
- **Balloon**: signaling messages are delivered from an initiator $i$ to a node $y$. They are intercepted by GIST nodes $z \in s_{iy}$; $y$ has to deliver the message to $c_y(r)$. Hence, signaling messages are received by all GIST nodes $z \in s_{iy} \cup c_y(r)$;
- **Hose**: signaling messages are delivered from $i$ to $y$. Each GIST node $z \in \pi_{iy}$ delivers the message to $c_i(r)$. Hence, signaling messages are received by all GIST nodes $k \in \bigcup_{z \in \pi_{iy}} c_z(r)$.

The definitions above could also make use of different metrics, such as GIST hops count. Without loss of generality, in what follows we will make use of the IP hop count.

Since the proposed off-path domains are built by using Bubble domains, we focus on the distribution of signaling messages within a Bubble. The off-path MRI for the proposed MRM is shown in Fig. 7. When an NSLP protocol requests the delivery of an off-path signaling message, it must indicate the metric specifying the radius of the off-path domain. Such a request is processed by the GIST initiator, which identifies the set of the destination peers compliant with this metric. The matching condition is verified for each element of the tuple $<$off-path domain type, metric type, value$>$\>. Before sending signaling messages to the selected destinations it is necessary to execute the GIST three-way handshake to create MA between peers and, if required by the NSLP protocol, the underlying transport protocol handshake\[18\].

For implementing the Bubble, we propose three algorithms for selecting the signaling destinations:

1) **GIST Flooding**: The flooding algorithm can make use of the information made available by either One-hop-based or Leaf-based discovery. The selection of destinations is done locally at each peer, as signaling spreads over the Bubble.

Each node receiving an off-path message reads the metric value $m$ in the MRI. If $m > 0$, the node searches in its PeT all the peers 1 GIST hop away whose IP distance is not larger than $m$. For each destination, it decrements $m$ by the IP distance of the selected next GIST hop, and then sends the message. Clearly, any forwarder does not send the message back to the sender. In order to avoid the packet duplication problem, typical of flooding protocols, we have introduced a new GIST error message that may be sent during the three-way GIST handshake. When a peer receives a GIST query, it stores the identifier of the served NSLP, the GIST session identifier, and the value of $m$ used to build the bubble, called $m_{prev}$. Then, if it receives a further GIST query from another peer with the same GIST session value, it replies with the error message and aborts the session if $m \leq m_{prev}$, otherwise it rebuilds the bubble by using $m$.

2) **GIST Broadcast**: This algorithm has already been proposed in [1], and it is used here as performance baseline. In this algorithm the destination selection is done by initiators, which aim to cover the complete set of nodes which meet the metric condition. An initiator $i$ identifies the sub-tree $\tau_i(r)$ included in the Bubble of radius $r$ (i.e. $\tau_i(r) = T_i \cap c_i(r)$), and sends a signaling message to each leaf of this sub-tree, denoting by $\lambda_i(r)$ this new set of leaves. Since each GIST node $i$ stores in its PeT all the discovered paths $s_{ij}$, in order to obtain $\tau_i(r)$ it is enough to truncate them by excluding the elements with a metric larger than $r$. The final element of each truncated path will provide the elements of $\lambda_i(r)$, discarding repetitions. If some paths to the destination peers share some GIST nodes, these nodes will receive replicated queries and data messages. However, in order to reach all the selected destinations, these nodes have to intercept and forward signaling messages without generating error messages.

3) **GIST Multicast**: In this algorithm, shown in Fig. 8, peers use the Multiple Destination field of the MRI shown in Fig. 7 in order to aggregate multiple signaling destinations within a single signaling session. An initiator $i$ identifies $\lambda_i(r)$ defined above, and initializes $l_i(r) = \lambda_i(r)$. Then, for each leaf $y \in l_i(r)$, it checks if $y$ shares the next-hop peer with any other leaf $z \in l_i(r)$. Any leaf $z$ matching this condition is inserted into the Multiple Destination field of the MRI which includes $y$ as destination address. All the leaves sharing the next-hop peer with $y$ are removed from $l_i(r)$, together with $y$ itself. The process goes on by grouping destinations until $l_i(r)$ is emptied. When the message is intercepted by a forwarder, the latter executes the same procedure, by using the destinations included in the MRI instead of computing $\lambda_i(r)$.

**Procedure GIST Multicast**

1: $l_i(r) = \lambda_i(r)$
2: for all $y \in l_i(r)$ do
3: $mri_{-ndest}_y = \emptyset$
4: for all $z \in l_i(r) - \{y\}$ do
5: if $s_{iy} \cap s_{iz} \neq \emptyset$ then
6: $mri_{-ndest}_y = mri_{-ndest}_y \cup \{z\}$
7: end if
8: end for
9: send message to $y$ with mult. dest. $z \in mri_{-ndest}_y$
10: $l_i(r) = l_i(r) - (mri_{-ndest}_y \cup \{y\})$
11: end for

**Fig. 8. GIST Multicast signaling destination selection.**

IV. **Experimental Results**

The performance of our proposals have been evaluated experimentally by using the testbed shown in Fig. 9, consisting...
of 60 nodes running Ubuntu Server v.10.04. Each node act as a backbone router in the points of presence (POPs) of an AS. An AS of 60 POPs is realistic [27]. Fig. 9 shows two types of nodes. Those labeled from S1 to S36 are network stubs. Also the node labeled as Tracker is a stub. Each stub is connected to one of the core nodes, labelled from C1 to C23. Our proposals have been implemented as an NSIS-ka extension.

We have implemented two overlay networks:
- **Full overlay topology**: All nodes execute NSIS.
- **Sparse overlay topology**: Core nodes labeled from C11 to C22 in Fig. 9 are legacy IP routers, not running NSIS. All the other nodes are NSIS compliant. This configuration is aimed at evaluating the impact of an incremental deployment of our extended NSIS.

### A. Network discovery results

The performance of the network discovery algorithms was evaluated by executing two sets of tests. The first one allowed evaluating performance during the transient phase, when all nodes are turned on. The second one was used to evaluate the overhead in the steady phase, when the GIST node discovery has been completed. We define the convergence time \( t_i \) of the node \( i \) as the time taken for executing the transient phase, that is when at least one complete gossip session has been executed with all NSIS nodes for the Leaf-based solution, and with all the NSIS neighbors for the One-hop-based solution, respectively. Hence, the network convergence time is \( \max_{i \in V} \{ t_i \} \). During the transient, we have evaluated also the overall network overhead.

We compared the network discovery performance of our proposals with that of the Random solution [1], where the PTG and PTS lists are randomly selected in the complete set of PEs stored in each node, using the Q-forward policy.

Fig. 10 shows the convergence time and signaling overhead during the transient phase as a function of \( H \), the number of shared PIs in a gossip message. In all experiments, differently from the Random discovery, both the Leaf and One-hop (labeled as “OH”) solutions provide satisfactory convergence time and signaling overhead. The label “OH, \( B = \infty \)" indicates the usage of a memory size larger than the number of PEs in the network, and the label “OH, \( B = 10 \)" indicates memory size of 10 PEs.

From Fig. 10a and Fig. 10c it emerges that when the Leaf solution is used both on sparse and full topologies, the convergence time does not decrease when more than 2 PIs are shared. This result is generally valid since sharing 2 identities is enough to fill the PeT of each peer with a sufficient number of uncontacted peers. This way, uncontacted peers can be selected as a PTG in most of subsequent cycles. By using the One-hop solutions, the convergence time increases with the number of shared identities \( H \). Sharing many peer identities makes the set of PEs, selectable for the next PTG, large. In addition, the number of possible PTGs is much higher than the number of next hop routers. When these routers are also NSIS nodes, it is disadvantageous to test a large set of PEs, most of which are unreachable due to the use of the Q-drop policy. When the paths to most of PTGs share the same next hop, the interception of other next hop routers happens less frequently. This process is even hampered by the presence of multiple IP addresses of GIST peers and by the small size of \( B \). In fact, multiple destination IP addresses could cause packets sent for contacting a neighboring peer passing through another GIST node, and the initiator would label this peer as unreachable. Only when the identity of this deemed unreachable peer will be associated with another of its IP addresses, the initiator will try to contact it again. The process goes on until the “right” IP address for that peer is communicated to the initiator. When the \( B \) value is lower than the number of NSIS nodes in the network, some unreachable nodes could be deleted from the memory, and, if they are shared again by a different peer, they are regarded as uncontacted peers and candidate for a new gossip session. Clearly, these operations can further increase the convergence time.

For the full overlay topology, we observe that when the number of shared peer identities is 1 or 2, the “OH, \( B = \infty \)” is preferable. In a sparse topology, the Leaf solution is preferable in all situations. Note that in a sparse topology the discovery time of the One-hop solution could slightly increase. In order to better highlight this behavior, Fig. 11a shows the discovery time of all implemented solutions, by black solid lines for the full topology, and by red dotted lines for the sparse topology. As expected, in the sparse topology the convergence time of the Leaf and Random solutions slightly decreases, due to a reduced number of NSIS nodes. Since the Leaf solution is designed to discover leaves, a significant decrease of the convergence time is expected when the number of leaves decreases. Instead, in the case of the One-hop solutions, while the implementation with \( B = 10 \) slightly benefits from the reduced number of NSIS nodes, such a result cannot be achieved when \( B = \infty \). In this case, a slight performance degradation is due to the increased average number of nodes being 1 GIST hop away, as it happens in the sparse topology. Fig. 10b and Fig. 10d show the signaling overhead during the transient phase in the full and sparse topologies, respectively. The overhead of both Leaf and Random solutions increases with the number of shared PEs, as the payload of gossip messages. In addition, when duration of the transient phase increases (Fig. 10a and Fig. 10c), the amount of exchanged NSIS traffic increases as well (Fig. 10b and Fig. 10d).

In a full topology, the overhead generated by the One-hop
solutions is significantly lower than that generated by the Leaf and Random solutions, since the One-hop solutions limit the scope of messages to 1 GIST hop, corresponding to this case to 1 IP hop. Differently, in a sparse topology, the average distance of the GIST nodes being 1 hop away increases, thus causing an increase of the overhead. In this case, the Leaf solution outperforms the One-hop. Leaf and OH solutions always outperform the Random one.

As for the steady state, we have both executed experiments and defined a theoretical model for the full topology. We denote by $\delta$ the IP network diameter, and by $\xi_{ij}$ the probability that a peer $i$ selects an LP $j \in L_i$ as a PTG. The length of the paths $p_{ij}$ in the PaT of $i$ is modeled as a discrete random variable distributed in the range $[p_i^{min}, p_i^{max}]$, with mean $\mu_i$ and variance $\sigma_i^2$. The overhead generated by the Leaf solution during a gossip session between nodes $i$ and $j$ is equal to:

$$\phi_{ij} = p_{ij}q + (r + a) \sum_{i=1}^{p_{ij}} i = p_{ij}q + (r + a) \frac{p_{ij}(p_{ij} + 1)}{2}, \tag{8}$$

thus, the average overhead generated by the $i$th node is:

$$\Phi_{i,Leaf} = q\mu_i + (r + a)\left(\frac{\mu_i^2}{2} + \frac{1}{3}(\mu_i^2 - \mu_i)\right), \tag{10}$$

it follows that:

$$\Phi_{i,Leaf} = q\mu_i + (r + a)\left(\frac{\mu_i^2}{2} + \frac{1}{3}(\mu_i^2 - \mu_i)\right). \tag{10}$$

Thus, the total network signaling overhead is:

$$\Phi_{Leaf} = \sum_{i=1}^{K} \Phi_{i,Leaf} = \sum_{i=1}^{K} \left[q\mu_i + (r + a)\left(\frac{\mu_i^2}{2} + \frac{1}{3}(\mu_i^2 - \mu_i)\right)\right]. \tag{11}$$

In order to provide an upper bound depending on global network parameters only (i.e. number of nodes $K$ and network diameter $\delta$), without requiring any knowledge of $\mu_i$, in (8) we replace $p_{ij} \leq \delta$ by $\delta$. Thus, it follows that:

$$\Phi_{Leaf,UB} = K\delta \left[q + (r + a)\frac{(\delta + 1)}{2}\right] \geq \Phi_{Leaf}, \tag{12}$$

which is a useful expression for evaluating the GIST overhead.

The overhead rate is found by dividing both $\Phi_{Leaf,UB}$ and $\Phi_{Leaf}$ by the gossip period $T_{gossip}$. An estimate of the bandwidth consumption during the transient phase is obtained by normalizing $\Phi_{Leaf,UB}$ and $\Phi_{Leaf}$ by the minimum value of $T_{gossip}$, and an estimate of the steady state bandwidth consumption is obtained by normalizing them by the maximum value of $T_{gossip}$. In fact, in the steady state PeT does not change, and thus $T_{gossip}$ increases up to its maximum value (see Fig. 4). The values of $minT_{gossip}$, $maxT_{gossip}$, and $maxCounter$ are equal to 5s, 40s, and 10 iterations, respectively. The length of Registration ($q$) and Response ($r$) messages is 156 bytes when a single PI is shared. It increases by 28 bytes for any further PI shared. The length of the ack message ($a$) is 112 bytes.

This model can be easily adapted to evaluate the mean overhead of the Random solution. It is enough to replace, in the equations above, the mean IP path length, $\mu_i$, with the mean IP distance, $\gamma_i$, from the node $i$ towards all other PEs.

For the One-hop solutions, we can provide a more general model. Let $\nu_i$ be the mean IP distance between the $i$th GIST node and its neighboring GIST peers. Since each peer executes a gossip session only with a neighboring peer, the resulting total signaling overhead is:

$$\Phi_{One-hop} = \sum_{i=1}^{K} \Phi_{i,One-hop} = \sum_{i=1}^{K} (q + r + a)\nu_i. \tag{13}$$

Fig. 10. Transient experimental results vs. size of the PTS list: (a) convergence time on full topology, (b) signaling overhead on full topology, (c) convergence time on sparse topology, and (d) signaling overhead on sparse topology. For One-hop (OH), $B=\infty$ indicates a memory size larger than the number of PEs in the network, whilst $B=10$ indicates a memory size equal to 10 PEs. 95% confidence intervals are shown.

Fig. 11. Full and sparse topology (a) convergence time and (b) steady phase overhead as a function of $H$. 

(a) Full Topology
(b) Full Topology
(c) Sparse Topology
(d) Sparse Topology

| Number of shared peer identities | Convergence time [s] | Signaling overhead [MB] |
|---------------------------------|----------------------|------------------------|
| Leaf, full top.                 |                     |                        |
| Random, full top.               |                     |                        |
| OH, B=10, full top.             |                     |                        |
| Leaf, sparse top.               |                     |                        |
| Random, sparse top.             |                     |                        |
| OH, B=10, sparse top.           |                     |                        |

| Number of shared peer identities | Convergence time [s] | Signaling overhead [MB] |
|---------------------------------|----------------------|------------------------|
| Leaf, full top.                 |                     |                        |
| Random, full top.               |                     |                        |
| OH, B=10, full top.             |                     |                        |
| Leaf, sparse top.               |                     |                        |
| Random, sparse top.             |                     |                        |
| OH, B=10, sparse top.           |                     |                        |
As in the full topology $\nu_i = 1$, the total signaling overhead is:

$$\Phi_{\text{One-hop, full}} = K(q + r + a).$$  \hspace{1cm} (14)

Fig. [12]a shows the experimental signaling bandwidth consumed by all strategies in the steady phase, as a function of $H$. For all solutions, bandwidth increases linearly with the number of shared identities. It emerges that the most efficient schemes are the One-hop ones, due to the limited scope of their signaling exchanges. The Random solution consumes less bandwidth than the Leaf one, since the PTG in the Random solution has a mean distance $\gamma_i \leq \mu_i$. However, the convergence time of the Random solution is higher than the one of the Leaf solution by one order of magnitude. In any case, considering the results relevant to the transient phase, which suggest to use $H = 2$, the total signaling overhead over the whole network is 70 Kbit/s, which is definitely negligible.

Fig. [12]b shows the signaling bandwidth consumption of the Leaf solution versus $H$, including theoretical results labeled as “theor.” [11], the upper bound labeled as “UB” [12], and the experimental results, labeled as “exp.” Results are shown for both the transient phase, labeled as “TP”, and the steady phase, labeled as “SP”. The first comment is that theoretical and experimental results of the steady state are in an excellent agreement, and also the rough upper bound is very close to the experimental results. As for the transient phase, the difference between theoretical and experimental results is due to the very initial phase of the discovery process. In this phase, not all exchanged PIs are LPs, thus some sessions span over paths shorter than those used in the steady state, thus making the overall signaling bandwidth lower than the theoretical one. The upper bound is about twice the experimental results, thus acceptable for a coarse estimation.

Fig. [12]c shows the theoretical and experimental signaling bandwidth of the Random solution as a function of $H$, in both transient and steady phases. Labels are the same as in Fig. [12]b. The agreement between the theoretical and experimental results in the steady state is excellent. For what concerns the transient phase, the overestimation of the theoretical model is due to the approach used for selecting the PTG and the PTS list by the Random solution. In fact, the mere random strategy could lead to stalls, in which a GIST node does not receive new PIs for several consecutive cycles and starts increasing $T_{\text{gossip}}$, thus decreasing the overall signaling bandwidth.

Fig. [12]d shows the theoretical and experimental signaling bandwidth of the One-hop solutions versus $H$, including both theoretical and experimental results for both the transient and steady phases. Differently from the Leaf and Random solutions, an excellent agreement between the theoretical model and experimental results for the transient phase is found, for both values of $B$. Instead, the theoretical model of the steady phase significantly underestimates the bandwidth consumption, which is close to the estimate for the transient phase. This is due to the management of $T_{\text{gossip}}$ in the One-hop solutions. In fact, differently from the other two solutions, multiple IP addresses of neighboring peers cause frequent changes in PeT. This problem could be avoided by sharing all IP addresses associated with a PI in each gossip message, but we did not proceed in this way since the resulting overhead would significantly increase. Thus, for most of the time, $T_{\text{gossip}} = \min T_{\text{gossip}}$. In addition, also the usage of a limited buffer could trigger a PeT change, and thus the usage of the minimum $T_{\text{gossip}}$ value. The use of a large buffer for storing unreachable peers produces small benefits, since the probability that a node increases its $T_{\text{gossip}}$ value is slightly higher. Thus, the overhead of the solution “OH, $B = 10$” is slightly worse than that of the solution “OH, $B = \infty$”.

Fig. [11]b shows the consumed bandwidth in the steady state, found experimentally, of all implemented solutions, in both the full and the sparse topologies. As expected, for sparse topologies, the bandwidth overhead of Leaf and Random solutions is lower. This is not true for the One-hop solutions. In fact, although $K$ decreases, the average IP distance between neighbors increases (see Fig. [13]). The resulting effect is an increase of the bandwidth overhead, even larger than the one generated by the Leaf solution.

In order evaluate the convergence time of the Leaf solution in larger networks, we made use of a custom Matlab simulator. We have generated 10 graphs, by using the model proposed in [28], with a number of nodes that ranges from 100 to 1000. The convergence time is expressed in number of gossip cycles. Fig. [13] shows the simulation results, a lower bound of the convergence time, obtained through the maximum number of leaves in a single tree $T_i$, $i \in \mathcal{V}$, and an upper bound, obtained by $K$. The distance between the lower bound and the Leaf
solution is due to the initial cycles of the algorithm, when PTS is selected from a small set of peers, and each node may share peers that are not leaves in its tree. Since these shared peers may be chosen by other peers in subsequent cycles, useless gossip rounds are executed thus slowing the algorithm down. In order to limit discovery time and overhead in very large networks, it is necessary to partition the network and limit the scope of each zone by setting suitable values of the IP TTL field for Leaf-based solutions. In summary, One-hop solutions are acceptable in some cases, but their performance is not easily predictable. For sparse topology, which is the most realistic one, the Leaf solution is preferable.

B. Off-path signaling results

Off-path signaling dissemination algorithms have been analyzed by using the SETUP message of the NetServ NSLP [3]. During the experiments we measured all the NSIS traffic at the IP layer, including both TCP and UDP protocols. We used two different configurations. In the first, the SETUP message was sent by a stub node (S13 in Fig. 9), while in the second it was sent by a core node (C2 in Fig. 9). The message was distributed over a Bubble with radius \( r \) ranging between 1 and 5 IP hops. Experiments have been repeated by using full and sparse topologies, with SETUP message sizes of 355 and 1024 bytes. Both values are typical of the considered NSLP. Fig. 14 shows the total signaling traffic generated by the off-path signaling experiments, using the full topology, as a function of \( r \). Each bar shows the amount of traffic generated by the transmission of a single SETUP message over the Bubble. The lower part of each bar is relevant to the TCP contribution. The GIST Broadcast algorithm generates significantly larger traffic than the GIST Flooding and the GIST Multicast ones, since queries (UDP), responses (UDP), confirm (TCP), and data messages (TCP), sent to LPs sharing common sub-paths, are received and re-transmitted more than once by the GIST nodes on common sub-paths. We refer to these GIST nodes as network forwarders, NFs. On the contrary, for both GIST Multicast and Flooding, each node processes NSLP data only once. Even though the presence of multiple destinations in the GIST header increases the GIST Multicast packets size, GIST Flooding increases traffic. In fact, in GIST Multicast the network initiator (NI) selects the nodes to be included in the off-path domain at the beginning of the signaling session. In GIST Flooding, the selection is done hop by hop by NFs, on the basis of their own PeT and PaT, possibly resulting in sub-optimal paths. In addition, the GIST Flooding algorithm generates a larger amount of UDP traffic due to error messages sent upon reception of duplicate queries. These differences are significant only for large \( r \) values, and, more importantly, when the NI is a stub and the message is small. In fact, when the message is small, the impact of duplicated queries and the relevant error messages is higher. In addition, when the NI is a stub, the bubble includes fewer nodes, thus the impact of these inefficiencies is higher.

As shown in Fig. 15 these differences increase in the sparse topology, since each node has a larger number of neighbors, with a larger IP distance between them. Hence, the GIST Flooding generates a lot of queries and error messages for each node involved in signaling. On the contrary, the GIST Multicast optimizes the distribution over the sub-trees of the overlay network, thus reducing the number of GIST hand-shakes. As \( r \) increases, it compensates the overhead generated by multiple destinations in the Multicast GIST packet header.

In summary, GIST Multicast and GIST Flooding are nearly equivalent for \( r \) up to 3 IP hops in all different configurations, and the advantages of the GIST Multicast algorithm become significant for a radius of 5 IP hops. The usage of randomized approaches, such as the Blind Broadcasting proposed in [29], would not reduce the additional traffic generated by the GIST Flooding algorithm, and they would largely increase the delivery time, especially in sparse topologies.

V. RELATED WORKS

The Generic Ambient Network Signaling (GANS) protocol [30] was proposed to establish signaling sessions having no direct relations with data flows. GANS is a GIST extension and allows dynamic interworking of heterogeneous networks. However, its role-based addressing scheme and its DEEP discovery protocol do not support multiple destinations [30].

The NTLP General Internet Service Protocol (GISP) [31] is a proposal which differs from GIST in the inclusion of signaling and transport services, whilst GIST makes use of existing transport and security protocols. In addition, GISP does not allow using different MRMs, thus resulting less modular and flexible than GIST.

Two NSIS extensions are proposed in [32]. They are aimed to introduce hybrid on-path/off-path capabilities for end-to-end signaling. The first makes use of the HyPath NSLP protocol, implemented as an intermediate layer between NSLP and NTLP layers. The second extends the GIST capabilities by introducing a TLV field in GIST messages, without using a new MRM, at a cost of a reduced flexibility.

A formalization of the gossip problem is proposed in [33]. Gossip solutions are mainly round-based and our solution makes no exception. Gossip rounds can be synchronous or asynchronous. Synchronous ones need of a synchronization
system that increases overhead. For this reason we proposed an asynchronous solution. Gossip protocols generally select peers involved in a gossip session randomly, but even gossip protocols using deterministic strategies exist \cite{34}. In our solution, PTGs are randomly selected, although heuristics are used for restricting the selection set. 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. This approach can be compared with gossip algorithms used in wireless multihop or ad-hoc networks. In \cite{35} the gossip protocol makes use of 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 GIST for sending gossip messages to multiple peers in a single round.

Gossip-based solutions have also been used to solve the discovery problem \cite{36} \cite{23}. Some analogies between our proposal and the one shown in \cite{36} exist, although the problem formalization is different. Both algorithms aim to create a network spanning tree, used for distributing messages. Nevertheless, the proposal in \cite{36} 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. In \cite{23}, the The two-hop walk is quite similar to the D-mode discovery process proposed in \cite{1}, but it assumes prior knowledge of the peer set of neighbors. A solution for collecting this information is proposed in \cite{37}.

We have improved our previous proposal \cite{1} in different aspects. During the bootstrap phase we have introduced two discovery strategies (Leaf and One-hop), that decrease convergence time by one order of magnitude in comparison to \cite{1}. We have also made possible to cope with asymmetric routing during the discovery phase. For what concerns the off-path distribution, we have introduced two algorithms (Flooding and Multicast) that definitely outperform the one presented in \cite{1}. An alternative bootstrapping solution consists of using an anycast address \cite{38}. This implies that a network administrator must allocate and maintain this special anycast address in all GIST nodes. Nevertheless, this solution would not support all the signaling MRMs of our proposal.

SNMP \cite{39} broadcast discovery could also be used to discover GIST nodes in a network, but its usage is currently discouraged for security reasons.

Even SLP (Service Location Protocol) \cite{40} and OSPF could be used to discover GIST nodes, for example through the use of OSPF capabilities of router advertising \cite{41} and link local signaling \cite{42}. Nevertheless, both of them are not suitable for our purposes. In fact, OSPF deals only with routers, and modifications to the routing daemons are often undesirable and discouraged by network operators. SLP works only in local area networks, where it makes use of Service Agents to inform User Agents about the available services through an extensive usage of multicast/broadcast traffic. In order to scale up to enterprise networks, it is necessary to both use Directory Agents and face all issues of a centralized architecture.

VI. CONCLUSION

In this paper we showed a proposal for providing path decoupled GIST signaling with minimal initial configuration. It addresses the problem in a fully distributed and automated way and generates a negligible overhead for modern networks. We have split the problem in two sub-problems: node discovery and signaling distribution. As for node discovery, the preferable solution, called Leaf-based, provides a stable convergence time, is flexible, and suitable for designing new and more complex distribution algorithms. We have also found an upper bound to network overhead, evaluated through general network parameters, such as its diameter and number of nodes. As for the distribution of path decoupled signaling, the preferable solution, called GIST Multicast, implements an efficient application layer multicast technique to “blow a bubble” of signaling messages around a network node. The GIST Multicast leverages the information provided by the Leaf discovery solution.

REFERENCES

[1] M. Femminella et al., “Gossip-based signaling dissemination extension for next steps in signaling,” in IEEE NOMS 2012, 2012, pp. 1022–1028.
[2] S. Guha and P. Francis, “An end-middle-end approach to connection establishment,” SIGCOMM Comput. Commun. Rev., vol. 37, no. 4, pp. 193–204, Aug. 2007.
[3] M. Femminella et al., “An enabling platform for autonomic management of the future internet,” Network, IEEE, vol. 25, no. 6, pp. 24–32, 2011.
[4] R. Van Renesse, K. P. Birman, and W. Vogels, “Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining,” ACM Trans. Comput. Syst., vol. 21, no. 2, pp. 164–206, May 2003.
[5] V. Shrivastava et al., “Application-aware virtual machine migration in data centers,” in INFOCOM, 2011 Proceedings IEEE, 2011, pp. 66–70.
Fig. 15. Off-path signaling overhead in sparse topology. Light color indicates TCP traffic, solid color indicates UDP traffic.

[6] V. Valancius et al., “Greening the internet with nano data centers,” in ACM CoNEXT ’09.
[7] W. Jiang et al., “Orchestrating massively distributed cnds,” in ACM CoNEXT ’12.
[8] N. Laoutaris, P. Rodriguez, and L. Massoulie, “Echos: edge capacity hosting overlays of nano data centers,” SIGCOMM Comput. Commun. Rev., vol. 38, no. 1, pp. 51–54, Jan. 2008.
[9] M. Yu et al., “Tradeoffs in cdn designs for throughput oriented traffic,” in ACM CoNEXT ’12, 2012, pp. 145–156.
[10] T. Wolf, “In-network services for customization in next-generation networks,” IEEE/ACM Transactions on Networking, vol. 24, no. 4, pp. 6–12, 2010.
[11] S. Shanbhag and T. Wolf, “Automated composition of data-path functionality in the future internet,” Network, IEEE, vol. 25, no. 6, pp. 8–14, 2011.
[12] A. Greenhalgh et al., “Flow processing and the rise of commodity network hardware,” SIGCOMM Comput. Commun. Rev., vol. 39, no. 2, pp. 20–26, Mar. 2009.
[13] D. Han et al., “XIA: Efficient support for evolvable internetworking,” in Proc. 9th USENIX NSDI, Apr. 2012.
[14] L. Popa, A. Ghodsi, and I. Stoica, “Htp as the narrow waist of the future internet,” in ACM Hotnets-IX, 2010.
[15] R. Hancock et al., “Next Steps in Signaling (NSIS): Framework,” IETF, RFC 4080, Jun. 2005.
[16] D. Katz, “IP Router Alert Option,” RFC 2113, Internet Engineering Task Force, Aug. 1996.
[17] H. Schulzrinne and R. Hancock, “GIST: General Internet Signalling Transport,” RFC 5971, Internet Engineering Task Force, Oct. 2010.
[18] H. Schulzrinne and R. Hancock, “GIST: General Internet Signalling Transport,” RFC 5971, Internet Engineering Task Force, Oct. 2010.
[19] Ns-is-ka, open source nsis implementation by kalsruhe university. [Online]. Available: [projecte.tm.uka.de/trac/NSIS/wiki](http://projecte.tm.uka.de/trac/NSIS/wiki)
[20] J. Manner, G. Karagiannis, and A. McDonald, “Nsis signaling layer (nslp) for quality-of-service signaling,” IETF RFC 5974, Oct. 2010.
[21] M. Steimerling et al., “NAT/Firewall NSIS Signaling Layer Protocol (NSLP),” RFC 5973, Internet Engineering Task Force, Oct. 2010.
[22] J. W. Lee et al., “Netsrv: Active networking 2.0,” in IEEE ICC 2011, 2011.
[23] B. Haeupler et al., “Discovery through gossip,” in ACM SPAA ’12, 2012, pp. 140–149.
[24] M. Jelasity, A. Montresor, and O. Babaoglu, “T-man: Gossip-based fast overlay topology construction,” Comput. Netw., vol. 53, no. 13, pp. 2321–2339, Aug. 2009.
[25] Ntp pool project. [Online]. Available: [http://www.pool.ntp.org/](http://www.pool.ntp.org/)
[26] T. H. Cormen et al., Introduction to Algorithms, 2nd ed. McGraw-Hill Higher Education, 2001.
[27] N. Spring et al., “Measuring isp topologies with rocketfuel,” Networking, IEE/ACM Transactions on, vol. 12, no. 1, pp. 2–16, 2004.
[28] A.-L. Barabasi and R. Albert, “Emergence of scaling in random networks,” Science, vol. 286, no. 5439, pp. 509–512, 1999.
[29] L. Gargano et al., “Efficient communication in unknown networks,” Networks, vol. 38, no. 1, pp. 39–49, 2001.
[30] N. Akhtar et al., “Gans: A signalling framework for dynamic interworking between heterogeneous networks,” in IEEE VTC 2006, 2006.
[31] T. Luu and N. Boukhkhemt, “Generic signaling service framework,” in IEEE CCNC 2005, 2005, pp. 290–295.
[32] L. Cordeiro et al., “Hypath: An approach for hybrid on-path off-path end-to-end signaling,” in IEEE GLOBECOM 2008, 2008, pp. 1–6.
[33] T. Gonzalez, “An efficient algorithm for gossiping in the multicasting communication environment,” Parallel and Distributed Systems, IEEE Transactions on, vol. 14, no. 7, pp. 701–708, 2003.
[34] J. Liu et al., “Deterministic gossiping,” Proceedings of the IEEE, vol. 99, no. 9, pp. 1505–1524, 2011.
[35] X. Gao et al., “An efficient heuristic gossiping mechanism in ad hoc routing,” in CHINACOM 2007, 2007, pp. 978–982.
[36] A. Bagchi, S. Hakimi, and E. F. Schmeichel, “Gossipping in a distributed network,” Computers, IEEE Transactions on, vol. 42, no. 2, pp. 255–256, 1993.
[37] A. Ganesh, A.-M. Kermarrec, and L. Massoulie, “Peer-to-peer membership management for gossip-based protocols,” Computers, IEEE Transactions on, vol. 52, no. 2, pp. 139–149, 2003.
[38] J. A. B. and K. Lindqvist, “Operation of Anycast Services,” RFC 4786 (Best Current Practice), Internet Engineering Task Force, Dec. 2006.
[39] J. Case et al., “Simple Network Management Protocol (SNMP),” RFC 1157 (Historic), Internet Engineering Task Force, May 1990.
[40] E. Guttman et al., “Service Location Protocol, Version 2,” RFC 2608, Internet Engineering Task Force, Jun. 1999.
[41] A. Lindem et al., “Extensions to OSPF for Advertising Optional Router Capabilities,” RFC 4970, Internet Engineering Task Force, Jul. 2007.
[42] A. Zinin et al., “OSPF Link-Local Signaling,” RFC 5613, Internet Engineering Task Force, Aug. 2009.

Mauro Feminella (M’01) received both the master degree and the Ph.D. in Electronic Engineering from University of Perugia in 1999 and 2003, respectively. Since November 2006, he is assistant professor at the Department of Engineering, University of Perugia. His research interests focus on nanoscale communications, signaling protocols, and middleware platforms.

Gianluca Reali is an associate professor at the University of Perugia, Department of Engineering, Italy, since January 2005. He received the Ph.D. degree in Telecommunications from the University of Perugia in 1997. From 1997 to 2004 he was researcher at DIEI. In 1999 he visited the Computer Science Department at UCLA. His research activities include wireless networks, network management, and multimedia services.

Dario Valocchi is a Ph.D. candidate in Information Engineering at University of Perugia, Italy, advised by Professor Gianluca Reali. His research interests focus on programmable networks and signaling protocols.

Roberto Francescangeli is Software Engineer at Schrodinger Inc. His research interests focus on distributed signaling architectures, advanced multimedia services, performance evaluation, and network management. He has been also responsible for the GENI integration of NetServ.

Henning Schulzrinne (M’86–SM’04–F’06) Levi Professor of Computer Science at Columbia University, received his Ph.D. from the University of Massachusetts in Amherst, Massachusetts. MTS at AT&T Bell Laboratories; associate department head at GMD–Fokus (Berlin), before joining the Computer Science and EE departments at Columbia University. He served as chair of Computer Science from 2004 to 2009 and as Chief Technology Officer of the Federal Communications Commission (FCC) from 2012. Protocols co-developed by him, such as RTP, RTSP and SIP, are now Internet standards, used by almost all Internet telephony and multimedia applications. His research interests include Internet multimedia systems, ubiquitous computing, and mobile systems.