A signaling protocol for service function localization

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

***Abstract*—Current proposals for chaining service functions (SFs) do not address some critical management issues, such as the discovery of SF instances close to IP data paths. This information is crucial for deploying complex services both in large cloud networks, where SFs may be moved or replicated, and in the emerging fog/mobile edge computing systems. For this purpose, in this letter we propose the distributed off-path signaling protocol (OSP). We show the protocol functions and demonstrate its scalability and effectiveness by experimental results.**

***Index Terms*—signaling, service chaining, gossip, off-path distribution.**

I. Introduction and Background

The service function (SF) management has recently been object of research, since it is strictly related to deploying complex services through the so-called network function virtualization (NFV) [1], by combining cloud computing and software defined networks (SDN). Many SFs can be virtualized, such as security functions, shaping, and caching. In this regard, the IETF Service Function Chaining (SFC) working group has defined an architecture based on SF chaining [2] to provide users with services by using virtualization and SDN [3]. All these activities include the management of network resources distributed over geographical networks, accessible in a virtualized environment [1]. However, SFs discovery, localization, and status retrieval are critical aspects not sufficiently considered in the technical literature, although it is explicitly mentioned in the ETSI specification [1].

SF instances in data centers could not be on the IP path of routers connecting two arbitrary communicating entities. Thus, in order to properly chain NFV instances, the use of SFs close to IP data paths could be essential for avoiding inefficient data redirections. Fig. 1 shows an example of SF chaining. The dashed blue arrow (Fig. 1.a) indicates the IP path. The localization of the available SFs allows selecting the suitable SF instances to identify the service path (dashed green arrow, Fig. 1.c), which, in turn, maps on the IP path followed by flows belonging to the SF chain (dashed red arrow, Fig. 1.b).

In small settings, a centralized orchestrator, knowing the whole network topology, could manage the localization and deployment functions. In large networks this approach in not scalable. In this case, a hierarchy of orchestrators could be used, but each of them would serve a portion of the network,

Manuscript received October 13, 2015; revised February 17, and April 19, 2016; accepted April 22, 2016. The associate editor coordinating the review of this letter and approving it for publication was Boris Bellalta.

M. Femminella and G. Reali are with Dept. of Eng., Univ. of Perugia, Perugia, Italy (corresponding author e-mail: gianluca.reali@unipg.it). D. Valocchi is with Dept. of Electronic & Electrical Eng., University College London, London, UK. This work is co-funded by EU under the project ARES, supported by GN3plus in the framework of the first GEANT open call.

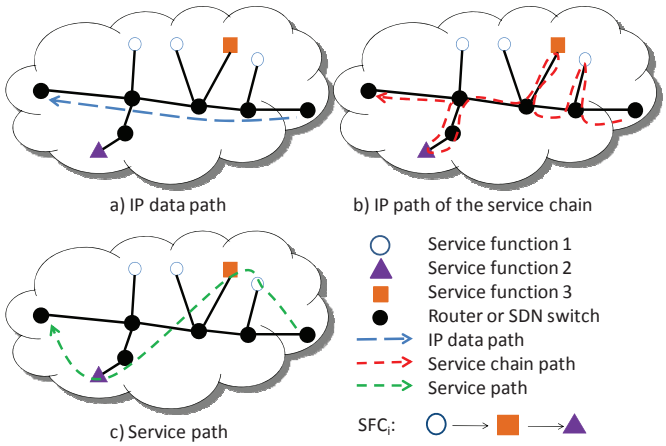

Fig. 1. SF chaining in large data networks: a) IP data path, b) IP service chain path, and c) logical service path.

and could take sub-optimal decisions. More importantly, multiple services could not be effectively managed. In fact, for this purpose controllers have to be constantly updated about the status of each service instance (e.g. each time a content is cached or evicted from a cache), and signaling congestion could occur. These effects would be even exacerbated both in the novel context of fog computing, which requires distributed management of mini-clouds at the network edge through NFV [4], and in mobile edge computing (MEC), an emerging technology for 5G networks, which includes base station softwarization [5]. The use of a localized and distributed management protocol for MEC/NFV would allow keeping SFs localized in edge clouds, with significant improvements.

In order to identify the suitable chain components and collect their status, we propose the off-path signaling protocol (OSP). It allows localizing resources close to IP data paths. OSP makes use of two main functions: on-path packet interception, used for data path identification, and off-path signaling [6], for SF localization with respect to a data path. Existing protocols do not include both features. For instance, the REsource LOcation And Discovery (RELOAD, IETF RFC 6940) protocol extends the Session Initiation Protocol by introducing peer-to-peer (P2P) off-path message exchange. Nevertheless, data path identification is not available. The Next Steps in Signaling (NSIS, IETF RFC 4080) protocol allows storing the state information on the NSIS peers lying on data paths, by leveraging its on-path packet interception capabilities, without supporting off-path signaling. Although an off-path patch was proposed in [7], it was not deployed since it inherits the complexity of the NSIS architecture.

OSP is illustrated in section II. Experimental results are

shown in section III. We draw our conclusion in section IV.

II. Off-path signaling protocol

The OSP architecture inherits some features of two existing solutions and avoids their shortcomings: NSIS and P2P gossip message exchange [8]. Although it inherits the on-path packet interception from NSIS for implementing off-path operations, its internal state management is highly simplified. Off-path signaling capabilities are protocol native functions. For their implementation, randomized gossiping mechanisms for peer discovery are used, as in P2P solutions [8]. The combination of path-coupled operations with off-path signaling allows identifying peers close to data paths. This function is not available in other P2P solutions. Our approach consists of first identifying nodes on the data-path, and then flooding signaling messages from each of them, in order to discover off-path nodes within a maximum distance from the data path. For this purpose, on-path interception is needed. It can be implemented easily, e.g. by port-based filtering or IP options, in SDN devices, software routers, or hardware routers with software development kit, thus in any carrier-grade networking platforms. OSP is organized in two layers. The upper layer, the Signaling Application (SA), implements the signaling logic, and provides a simple interface to the NFV management application. The lower one, the Signaling Transport (ST), distributes SA messages to the intended recipients. In what follows, we describe how these functions are implemented.

A. The peer discovery in the Signaling Transport layer

Peer discovery is a preliminary function used to fill peer tables (PeTs), which are ST data structures used in the off-path signaling distribution described in Section II.B. They include the identity of neighboring peers and the measured IP distance and latency. Each ST node stores in its Peer Table (PeT) the unique peer identifier (PID), the peer IP address, its IP hop distance, the estimated round-trip time, a timestamp of the last gossip session, and a flag indicating its reachability.

OSP is asynchronous and round based. When an ST node is turned on, it only stores the identity of a default node (tracker), used to obtain an initial set of peers. An ST node periodically gossips with the known peers for obtaining further ST node identities, acting as Gossip initiator, by sending *Gossip-Registrations*. To limit the network overhead, the reachable range of the gossip exchange is set to 1 ST hops. *Gossip-Registration* messages are intercepted and dropped by the first ST node on the path (the Gossip responder) toward the original destination. It replies back with a *Gossip-Response*, which is followed by a final *Gossip-Ack*. Registrations and responses include the identities of some other peers (the peer to share list, PTS), randomly selected from the PeT of the initiator/responder, as it typically happens in gossip protocols, such as the Newscast protocol [8].

The protocol operation is illustrated in Fig. 2, which shows also the evolution of PeTs. The initiators are N1 (two times), and then N3. For example, when the first initiator N1 receives the *Gossip-Response*, it stores the identity of the responder N3 in its PeT, along with the relevant measured metrics. The flag in its PeT entry is set to 1, which indicates that the responder

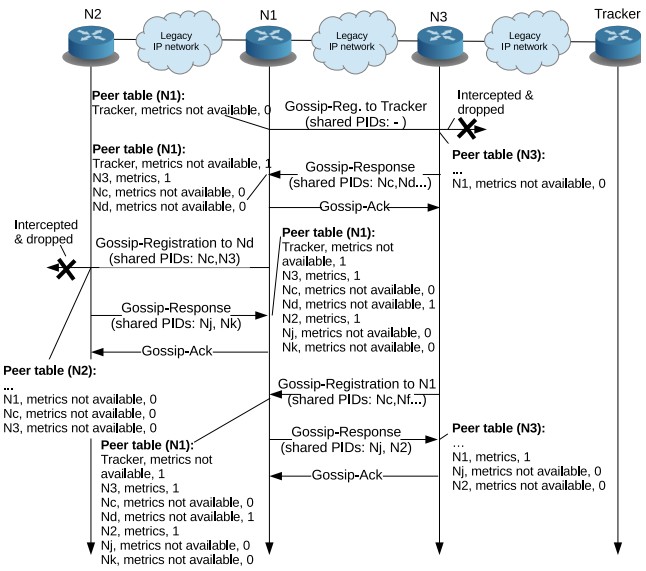

Fig. 2. Evolution of the gossip-based peer discovery at ST layer.

has been contacted. In case an ST node (responder) intercepts the *Gossip-Registration*, the flag associated with the original destination (Tracker in Fig. 2) is set to 1 and its relevant metric values are set to a non-significant value (e.g. -1), i.e. it means that the destination is out of scope. The *Gossip-Ack* notifies the responder that the initiator has received its PTS. By using the information in OSP and IP headers, it is very easy to evaluate the IP distance between initiator and responder. Each peer identity received for the first time in the PTS, or an intercepted, previously unknown, Gossip initiator, has the flag temporarily set to 0 (uncontacted, i.e. a not valid metric). N1 is initially set as uncontacted in the PeT of N3. The selection of the destination of the next gossip cycle is random, with a higher priority for peers whose flag is 0. Each peer in the PeT is associated with a lifetime to cope with the transient nature of virtualized SFs. If a *Gossip-Registration* gets no answers, the relevant peer is set *out of scope*. See [9] for further details.

B. The signaling distribution

The distribution function is managed jointly by the two layers, ST and SA. Communication primitives confine transport functions at the ST layer, and decision logic at the SA one. Fig. 3 shows their finite state machines (FSMs). In these diagrams, transition edges are labeled with the triggering event (above) and the triggered actions (below). Both SA initiator and forwarder behaviors are modeled by using three states: IDLE, Wait Notification, and Wait Responses. This state definition is flexible enough to both integrate NFV instances easily and introduce multiples SAs protocols. The ST FSM is slightly more complex, since it has to deal with additional lower layer issues, including packet interception and peer selection for signaling distribution. As for the distribution, we consider an off-path domain which includes the ST nodes staying within a maximum distance r (off-path radius) from at least one of the nodes of the IP data path. In this paper, we use the IP hop count metric. At the ST layer, the off-path signaling

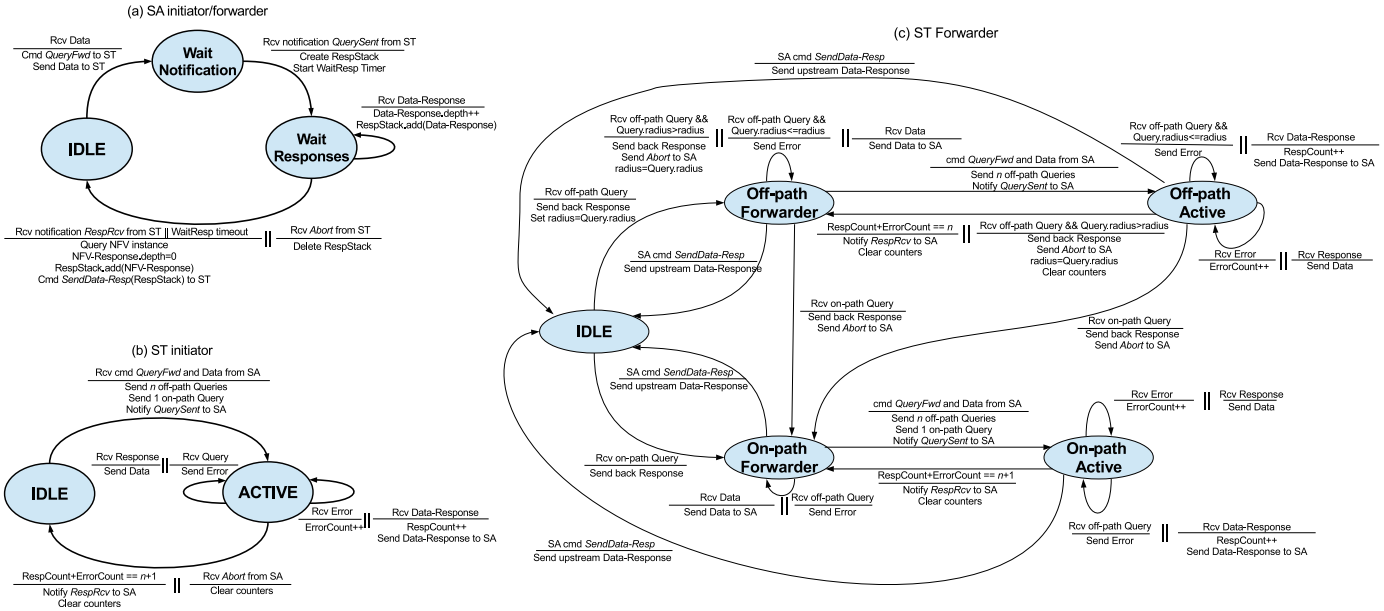

Fig. 3. State machine for the (a) SA initiator/forwarder, (b) ST initiator, and (c) ST forwarder.

distribution adopts a flooding algorithm, which makes use of two sets of peers: those laying on the IP path between the signaling initiator and destination (on-path peers), and those laying within a distance of radius r IP hops from the path (off-path peers).

1) Signaling delivery: downstream: The signaling exchange is initiated by the SA, triggered by an external application (transition from IDLE to Wait Notification in the SA FSM of the initiator, Fig. 3.a). This action triggers a command sent to the ST layer of the initiator (transition from IDLE to ACTIVE in Fig. 3.b). The latter generates an initial query message, by setting both the desired value of the radius r and a specific *on-path flag* in the message header of the on-path query, and sends it to the signaling destination. This flag is marked in the signaling messages delivered on the IP data path. Queries are intercepted by the other ST nodes. Each OSP node receiving an on-path query must accept the peering request, send a response message, and be ready to receive SA data from the upstream node (transition from IDLE to On-path Forwarder in Fig. 3.c). When these data are received, they are forwarded to the SA (loop on On-path Forwarder in Fig. 3.c and transition from IDLE to Wait Notification in Fig. 3.a). Then, the SA layer triggers the ST layer to deliver the signaling message not only on-path, but also to the n neighbors with a metric value $d \leq r$, namely off-path ST nodes. For each selected off-path peer i in PeT with $d_i \leq r$ (Fig. 2), the ST node generates a new query with an updated radius $r - d_i$ and the on-path flag set to 0 (off-path queries). The new queries are then sent to all the selected peers. The upstream ST node is not selected for off-path distribution. Then, the ST layer notifies the SA and performs a transition towards the Active state (On-path or Off-path, Fig. 3.c), waiting for responses from the queried peers. In turn, the SA FSM moves into Wait Responses (both Fig. 3.a), and creates a stack data structure to store data responses, which are expected within the responses traveling back towards the initiator.

At the ST layer in the Active states, when positive responses are received by the queried peers, SA data are delivered to them (loops on the Active states in Fig. 3.c and on the ACTIVE state in Fig. 3.b). When a node receives an off-path query, it reads the value of r. If $r \geq 1$, the procedure illustrated above is used to select the signaling destinations. The ST state transition is from the IDLE to the Off-path Forwarder (Fig. 3.c). For avoiding the packet duplication problem, typical of flooding algorithms, we have introduced an ST error message. When a forwarder receives a signaling message, it creates an internal soft state for the signaling session, storing the identifier of the served SA protocol, the session identifier, the upstream PID, and the value of r. Then, if it receives another ST off-path query from another peer before the timeout, with the same set of values and radius $r' \leq r$, it rejects the peering request and sends back an error message (error loops on all states except IDLE in Fig. 3.c and Fig. 3.b). In addition, the ErrorCount variable, set to 0 at session setup, is incremented. Instead, if $r' > r$, the previous session is aborted, the SA is notified (transition to IDLE in Fig. 3.a), and a new session is created at ST (transition to Off-path Forwarder, Fig. 3.c).

Similarly, when an off-path node receives an on-path query, it aborts the previous session and establishes a new one, acting as an on-path node. In fact, the value of the radius r for an on-path node is always the maximum one, that is that selected by the NFV application that triggers the signaling distribution. In this case, the SA is notified and the relevant states are deleted. The relevant transitions shown in Fig. 3.c are those from off-path states to On-path Forwarder.

2) Signaling delivery: reverse path: When a node is a final destination of the off-path signaling and cannot forward the message further, the ST layer moves to the Off-path Active state just to notify the SA layer and then returns into the Off-Path Forwarder state. The SA queries the local NFV instance to get local data, pushes the response into the stack with a depth parameter equal to 0, and triggers the ST to transmit the data response upstream. Session state at SA is cleared, and the FSM returns in IDLE, and the same is done at the

ST layer, after sending the data response upstream. When the ST of an intermediate forwarder receives a data response, it increments the local counter RespCounter, initialized to 0 at session creation, and passes the data response to the SA layer through the relevant APIs (loops on Active states in Fig. 3.b and Fig. 3.c), by also including the metric value d of the sending peer, taken from the PeT. The SA pushes this data response into the stack prepared upon entering the Wait Response state, and increases its depth values (loops on Wait Responses state in Fig. 3.a).

When all the expected responses are received by the ST layer, which means that the number of responses and error messages is equal to n, the ST returns back into the Forwarder state in Fig. 3.c, and notifies the SA. The SA queries the local NFV instance, pushes the data into the stack with a depth value equal to 0, and triggers the ST to send the data response stack upstream. Then, it clears all the state variables and moves in IDLE. In turn, the ST sends these data upstream, clears all state information, and moves in IDLE as well. If the WaitResp timer expires at SA, the same actions are executed, without waiting for all the notifications from the ST. If these events occur at the SA initiator, it sends the collected responses to the querying application going to IDLE.

III. Performance Evaluation

OSP has been analyzed through real experiments. The testbed emulates the Geant network topology, using Linux virtual machines (VM) running in a Gigabit Ethernet cluster. The topology includes 41 routers and 32 servers, used to model points of presence and datacenters, respectively, each implemented by a VM with an OSP instance. We analyzed peer discovery and signaling distribution of OSP.

The peer discovery analysis, which runs in background (Section II-A). The time needed by each node to discover all its neighboring peers is denoted *gossip discovery time* (T_{GD}). The best value of T_{GD} is achieved with a PTS list of 2 peers, resulting in a mean number of gossip cycles (n_{GC}) equal to 36, with $T_{GD} = n_{GC} \times T$, where T=5s is the gossip period. This value is about 16 times lower than the T_{GD} of the GIST solution in [7]. The minimum OSP n_{GC} value is the maximum degree of the OSP overlay (10 in the used topology), whereas the maximum OSP n_{GC} is the number of peers of the overlay, except the peer itself (K-1=72). As for the overhead, since at each gossip cycle the ith OSP node carries out a complete gossip query/response/ack session with one of its neighboring peers, then the total average bandwidth overhead is $\eta = \frac{1}{T} \sum_{i=1}^{K} \nu_i \, (G + R + A)$, where G=184 bytes, R=184 bytes, and A=112 bytes are the size of registration, response, and ack gossip messages, respectively, and ν_i is the mean IP distance between the ith OSP node and its neighboring peers. With the selected short gossip period (worst case), the OSP gossip discovery produces a negligible signaling bandwidth equal to 55 Kbit/s for the whole network (a fraction equal to 3×10^{-7} of the whole network bandwidth), when $\nu_i = 1$, that is OSP is deployed in all network nodes. Additional details can be found in [9].

We now consider the signaling distribution triggered by an NFV application (Section II-B). In all experiments, with all neighbors discovered (worst case), the signaling delivery times are less than 1 s for any path length. Fig. 4 shows the aggregated overhead generated by the signaling distribution as a function of the IP data path length (L), for increasing values of r. We compare OSP with GIST [7], since other gossip algorithms, such as Newscast [8], do not have the requested features (data path proximity control). We measured the overhead at IP layer, and averaged results over different pairs of peers with the same IP distance. The overhead increases with path length and domain radius. In the worst case (L=9, r=3), the delivery of a signaling message (size 1KB) to a large set of nodes generates only 200 KB of traffic over the whole network. Since a session is completed within 1s, it requires a fraction of the whole network bandwidth equal to 9×10^{-6}, which is negligible. OSP definitely outperforms GIST [7], with an improvement of about 30-40% for r=1, 6 times for r=2, and 11-14 times for r=3.

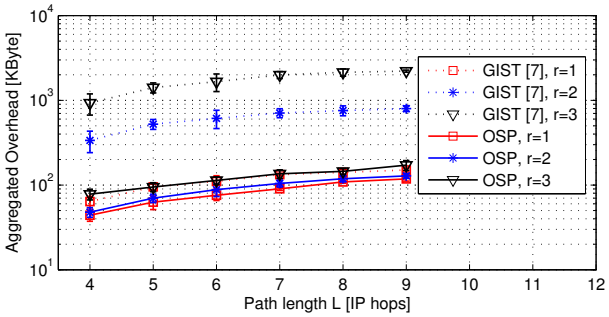

Fig. 4. Network overhead for NFV probing over an off-path domain of size r (IP hops) vs. IP path length L. Bars indicates 95% confidence intervals.

IV. Conclusion

This paper illustrates a new signaling protocol, called OSP, for discovery and localize service functions and make them available for chaining. The original feature of this protocol is its off-path scope, which is enabled through gossip-based discovery and flooding-based distribution. OSP has been implemented and analyzed experimentally. It exhibits the desired features at the expenses of a negligible overhead. Future work will consider integration with MEC platforms in 5G networks.

References

[1] "Network functions virtualisation (NFV); management and orchestration," ETSI GS NFV-MAN 001 V1.1.1, Dec. 2014.
[2] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *IEEE CloudNet '14*, Oct 2014.
[3] P. Quinn and J. Guichard, "Service function chaining: Creating a service plane via network service headers," *Computer*, vol. 47, no. 11, Nov 2014.
[4] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, Oct. 2014.
[5] Y. C. Hu et al, "Mobile edge computing - A key technology towards 5G," ETSI White Paper No. 11, Sept. 2015.
[6] S. Guha and P. Francis, "An end-middle-end approach to connection establishment," in *SIGCOMM '07*.
[7] M. Femminella, R. Francescangeli, G. Reali, and H. Schulzrinne, "Gossip-based signaling dissemination extension for next steps in signaling," in *IEEE/IFIP NOMS'12*, April 2012.
[8] S. Voulgaris, M. Jelasity, and M. van Steen, "A robust and scalable peer-to-peer gossiping protocol," in *AP2PC '03*, 2005.
[9] M. Femminella, G. Reali, and D. Valocchi, "A signaling protocol for service function localization - Supporting document," 2016. [Online]. Available: http://arxiv.org/abs/1604.08357