Distributed connectivity service for a SIP infrastructure

Luigi Ciminiera, Guido Marchetto, Fulvio Risso, Livio Torrero
Politecnico di Torino, Dipartimento di Automatica e Informatica
Corso Duca degli Abruzzi 24, 10129 Torino, Italy
{lui.ug.ciminiera, guido.marchetto, fulvio.risso, livio.torrero}@polito.it

Abstract—Because of the constant reduction of available public network addresses and the necessity to secure networks, devices like Network Address Translators and firewalls have become quite common. Being designed around the client server paradigm, they break connectivity when protocols based on different paradigms are used (e.g., VoIP or P2P applications). Centralized solutions for middlebox traversal are not an optimal choice because they introduce bottlenecks and single-point-of-failures. To overcome these issues, this paper presents a distributed connectivity service solution that integrates relay functionality directly in user nodes. Although the paper focuses on applications using the Session Initialization Protocol (SIP), the proposed solution is general and can be extended to other network scenarios.

I. INTRODUCTION

While end-to-end direct connectivity was a must in the early days of the Internet, nowadays an increasing number of hosts is connected through middleboxes such as Network Address Translators (NAT), which enable the reuse of private addresses, or firewalls, which are used to secure corporate networks and internal resources, or even both. These devices work seamlessly in case of client-server applications (although the client must reside in the “protected” part of the network), but they limit the end-to-end connectivity of all the applications that use different paradigms. Particularly, middleboxes prevent nodes behind them to be directly contacted from external nodes, e.g., for establishing a multimedia session. Various strategies have been proposed for traversing these middleboxes. The mostly used are hole punching and relaying [1], which are deployed especially for NAT traversal. The common idea is to make the middlebox believe that the initiator is always the internal host: in this way, the middlebox will create a temporary binding with the remote host, thus allowing the delivery of packets coming from the outside world. Particularly, the hole punching implements this idea by forcing each internal host to keep a persistent connection with an external node (the relay server) which operates as a forwarder, i.e., it receives all packets directed to the internal host and redirects them to it. This solution requires that the internal host advertise the IP address of the relay server as one of its address, and that instructs the relay server with the proper forwarding rules.

This paper focuses on the problem of middleboxes traversal for applications using the Session Initialization Protocol (SIP) [2] as it is one of the protocols that are mostly sensitive to the presence of middleboxes limiting the connectivity of end-users. Two solutions have been defined for NAT traversal in this context (in SIP, firewall traversal is a topic of minor interest, although the two problems are similar). The delivery of SIP messages relies on a relay mechanism, which consists in forcing all messages to be forwarded to a public host (the SIP proxy), and from here to the destination SIP User Agent (UA) [3]. For media flows, the Interactivity Connectivity Establishment (ICE) [4] protocol has been proposed, which implements a hole punching-based approach, with a relay-based approach as a fallback. These proposals rely on centralized servers acting as rendez-vous and relays nodes, but this lacks in robustness as the centralized server is a single point of failure: if the server fails, all UAs behind middleboxes become unreachable. Furthermore, a centralized solution cannot scale to an IP-based telecommunication provider with millions of customers: the server would handle an amount of traffic (both SIP signaling messages and media datagrams) that can be significant [5], thus requiring a high amount of computational resources.

This paper proposes a distributed architecture — referred to as DIStributed COmnectivity Service (DISCOS) — for ensuring connectivity across NATs and firewalls in a SIP infrastructure. This solution overcomes the limitations of the current centralized solution by creating a gossip-based P2P network and integrating rendez-vous and relay functionalities in each UA. Each globally reachable UA with enough resources can provide such services to UAs with limited connectivity, offering either hole punching or, if this fails, relaying. Although SIP has P2P extensions defined in P2PSIP [6], the latter is mainly a solution for distributed lookup whereas DISCOS offers a solution for middlebox traversal, which (to some extent) are orthogonal functionalities. In addition, this protocol has been engineered...
and validated by simulation on a SIP infrastructure, but the solution is more general and it can be seen as a mechanism to cope with middlebox traversal, thus opening the path to a wider adoption.

The idea of distributing such functionalities among end-systems is also one of the characteristics of Skype, a well-known VoIP application. However, Skype uses secret and proprietary protocols which cannot be studied and evaluated by third parties, therefore limiting the ability to understand exactly how these problems are solved and to evaluate the effectiveness of these solutions. For instance, the Skype blackout in August 2007 could be due to a failure of these mechanisms.

II. OPERATING PRINCIPLES

A. Distributed Connectivity Service

The Distributed Connectivity Service (DISCOS) replaces standard centralized solutions that current SIP infrastructures offer to ensure connectivity across NATs and firewalls. In DISCOS, a UA with enough resources (e.g., a public network address, a wideband Internet connection and free CPU cycles) becomes what we define a connectivity peer and starts to offer connectivity service (i.e., rendez-vous and relay functionalities). UAs with limited connectivity can locate and attach to an available peer whenever they need a relay. In a SIP infrastructure, UAs need one to either receive SIP messages (i.e., a SIP relay) or have support for media sessions when hole punching fails (i.e., a media relay). SIP relays can also offer support to the hole punching procedure for media session establishment, thus operating as a distributed rendez-vous server.

Connectivity peers are organized in a P2P overlay. This solution makes the transition from a centralized service to a distributed one seamless for UAs with limited connectivity as it provides an easy manner to locate a connectivity peer. The knowledge of connectivity peers is spread through proper advertisement messages, thus building an unstructured gossip-based network. In unstructured overlays, each node stores the addresses learnt in an internal cache. No strong relationship among nodes is needed, differently from structured overlays like the Distributed Hash Tables (DHT) that are based on particular routing tables located in each node. Structured networks are the best solution for lookup of specific resources, but this is achieved at the expense of an additional overhead due to the maintenance of the above mentioned tables. However, there is no need for such lookup properties in DISCOS, where the overlay is just used to find the first available connectivity peer: we have only one resource (the connectivity service) provided by all nodes of the network.

B. Overlay Topology

In order for DISCOS to provide an available peer to UAs with limited connectivity, possibly in the shortest time, peers should have a deep knowledge of the network: the greater is the number of known peers, the higher is the probability of finding an available peer in a short time, especially if known peers are lightly loaded. In gossip-based networks, the spread of information is based on flooding, thus the overlay topology has a deep impact on the network efficiency. For instance, the greater is the number of known peers, the higher is the probability of

The depth of the flooding (hence the load on the network) that is needed for an adequate spread of the information, with possible overlay collapsing. Thus, an overlay topology that ensures a small average path length is required. However, this is not sufficient for enabling peers to know a large set of suitable connectivity peers, from which to choose when a UA asks for the connectivity service. Nodes maintains a cache which may be kept small in order to reduce the overhead required to manage all the entries, thus limiting the number of peers known in each time. The limited cache size can be compensated by frequently refreshing its contents, so that the set of known peers changes frequently, resulting in a sort of round robin among peers: always changing connectivity peers can be provided to UAs that ask for the service at different instants, thus increasing the opportunity for a queried connectivity peer to suggest available ones when it cannot provide the service. Frequent cache refresh is also useful for ensuring that nodes store up to date information about existing peers. Such policy can be efficiently adopted if the overlay results in a scale-free network [7], an interesting topology that ensures small average path length and features scalability and robustness. In a scale-free network, few nodes (referred in the following as hubs) have a high degree while the others have a low one. The degree of a node is the sum of all its incoming (i.e., the in-degree) and outgoing (i.e., the out-degree) links. In the DISCOS overlay, the out-degree of a node is limited by the cache size while the in-degree is the number of other peers that have the node in their cache. Thus, nodes can be considered hubs when they are in the cache of lots of peers, i.e., when they are highly popular. Being hubs highly popular, they can discover lots of connectivity peers as they frequently receive advertisement messages from a large set of different nodes. In particular, if messages contain lowly popular nodes, hubs can discover peers that, being lowly popular, are lightly loaded with high probability. The key is to make searches through hubs since they potentially know a large variety of lightly loaded peers. Thus, the proposed solution essentially exploits — and generalizes to the case of a single resource provided by many nodes — the results achieved by Adamic et al. [8] about random walk searches in unstructured P2P overlays. They demonstrated that searches in scale-free networks are extremely scalable (their cost grows sublinearly with the size of the network), proving also that searches towards hubs perform better than random searches, since hubs have pointers to a larger number of resources. In DISCOS, as described above, the benefit of searching through hubs comes from the high frequency with which pointers to connectivity peers change in their cache. These properties are obtained to the detriment of a non-uniform distribution of the number of messages handled by nodes: the higher is the popularity of a node, the greater is the number of advertisement messages it
receives. However, a proper hub selection policy and a reasonable advertisement rate could mitigate the effects of this disparity. These aspects will be better analyzed in the following section.

The Barabasi-Albert [7] model has been proposed to create scale-free graphs. In this model, few nodes are immediately available and, when a new node arrives, it connects to one of the existing nodes with a probability that is proportional to the degree of such node (preferential attachment); in other words, the model assumes a global knowledge of nodes and their degree, which is clearly inapplicable in a real network scenario. A first step to implement such model in our overlay is to force $M$ nodes to register as connectivity peers in a DISCOS Bootstrap Server. When new nodes want to join the overlay, they query the Bootstrap Server for a subset of these $M$ registered nodes. However, preferential attachment is not possible with the mechanism described so far because all incoming peers $(i)$ can learn only the nodes present in the Bootstrap Server and $(ii)$ cannot compute the popularity of a node. An adequate spread of the network knowledge can address the first issue, but there are no ways to enable a node to learn the in-degree (i.e., the precise metric of node popularity) of the others. In our case, a simple approximated metric for the popularity of a node is the number of advertisement messages that contain such node. The basic idea to implement preferential attachment in our approximated model is to force peers to evaluate popularity of nodes through the above mentioned mechanism and then to include some most popular peers in the advertisement messages they send. This allows receiving nodes to put highly popular peers (hubs) in their cache, thus building and maintaining the scale-free topology. In summary, entering nodes use the peers retrieved from the DISCOS server as “bootstrap” nodes, then they learn most popular ones through the received advertisement messages and start to perform preferential attachment.

C. Protocol Overview

According to the above described overlay construction model, whenever a UA joins the overlay, it fetches a subset of the registered peers from the Bootstrap Server. First, it performs a validation phase to determine if it can become a connectivity peer or it is an UA with limited connectivity, using the rendez-vous functionalities provided by the overlay.

The protocol diverges according to the status of the UA, i.e. if it is an UA with limited connectivity or not. If the UA can become a connectivity peer, it checks the number of addresses registered on the Bootstrap Server and if it is smaller than a fixed bound $M$, it adds itself to the list. The UA inserts the just learnt peers in its cache and sends an advertisement message to them in order to announce itself. Then, it starts to receive messages coming from other nodes, thus gradually filling its cache with new peers. A proper peer advertisement policy is adopted in order to both implement preferential attachment (as described in our overlay construction model) and enable caches to be refreshed with lightly loaded peers. In particular, advertisement messages include the sender, the two most popular peers it knows (in order to allow recipients to implement preferential attachment, thus building and maintaining the scale-free topology), and two less popular peers it knows (with the purpose of refreshing caches with lightly loaded peers).

Advertisement messages are periodically sent by peers to all nodes they have in their cache and contain a special TTL field that allows the message to cross N hops: as soon as the message is received the TTL value is decremented and, if it is greater than 0, the recipient sends another message to all the nodes in its cache. Every time a peer receives an advertisement message, it updates its cache by increasing the popularity of nodes already present and by inserting the new ones. As previously described, it is important for a node to have both hubs and lowly popular peers in its cache: the knowledge of hubs implements the scale-free topology, while the knowledge of lowly popular peers (especially for hubs) allows the node to know unpopular (and hopefully) lightly loaded peers to UAs that ask for the connectivity service. Thus, also a proper cache management policy is adopted if the cache is full: the node with average popularity is removed before the insertion, resulting in a cache that privileges both big hubs and lowly popular peers.

Ad-hoc messages are also periodically sent to the UAs with limited connectivity to which a peer is providing the connectivity service. The purpose of these messages is described in the following.

UAs with limited connectivity have a different behavior. In this case, the list of peers obtained by the Bootstrap Server are stored in the UA cache and used to determine a list of potential connectivity peers. These peers will be contacted with a special message with the purpose of selecting a set of potential relays. The contacted connectivity peer may accept or refuse the request. If it refuses, it includes in the answer the two less popular peers and the most popular peer it knows: the less popular peers are queried immediately (since they are supposed to be free enough to provide connectivity), while the most popular is inserted in the cache (since it can perform faster search as potentially it is a hub). If both the queried peers refuse to provide the service, another node is picked from the cache and the procedure is repeated. If all the nodes in the cache have been queried without success, two different policies are applied depending on the type of service the UAs with limited connectivity needs: in the case of lookup for a SIP relay, the UA waits for a random time and then repeats the procedure; in the case of lookup for a media relay the procedure is stopped and the media session is considered blocked.

When a UAs with limited connectivity identified a suitable relay, it starts to send keep-alive messages in order to maintain the communication channel with the relay itself. The traffic generated by these messages is limited because each connectivity peer serves a limited number of UAs and, more important, is distributed within the network, while in case of the current centralized solutions implemented in SIP, messages converge to the same central servers.
As mentioned above, UAs with limited connectivity periodically receive messages from their relays. These messages contain three highly popular peers and thus allow UAs with limited connectivity to fill, first, and then update, their cache with new hubs. This enables them to direct searches towards hubs when they need for a connectivity peer (e.g., when their current SIP relay goes down or when they need to find a media relay to perform a media session). Broken hubs (e.g., because of a network failure) are detected through a timeout: if an hub does not reply to a query, UAs with limited connectivity can query others hubs they have in their cache. If no hubs are available, the UA fetches again the registered peers from the Bootstrap Server, although this situation is unlikely to occur as UAs with limited connectivity periodically receive new hubs from their SIP relays.

III. OVERLAY SIMULATION

Simulations are run to evaluate the proposed solution. Firstly, effectiveness and scalability are considered. Then, load balancing among peers and media session failure probability are discussed. This section reports on simulation results.

A. Simulations background

Statistics presented in [10] show that about 74% of hosts are behind NAT. In addition [1] shows that hole punching is successful in about 82% of cases. To the best of our knowledge, no detailed information is available about firewall proliferation over the Internet. On the strength of these available data, we consider for simulation a network scenario where 74% of nodes have limited connectivity and 18% of sessions coming from these nodes need relaying also for media sessions. TTL field of the advertisement messages is set to 2, sending interval of such messages is 60 minutes, and the cache contains 10 entries. Furthermore, the number of peers registered in the Bootstrap Server is set to 20. To approximate as closely as possible the behavior of real VoIP networks, node lifetime and call length statistics have been modeled after analyzing Skype traffic coming from/to the network of the University campus, while node arrivals are modeled using a Poisson process. With our parameters, the average number of nodes in the network depends on their arrival rate, because of the effect of Poisson arrivals model coupled with Skype’s lifetime distribution. For example, an arrival rate \( \lambda_N = 100 \) nodes/minute leads to a network consisting, on average, of 30000 nodes, which is the standard size in our simulation and it is a good trade-off between simulation length and significance of results. Occurrences of media sessions are modeled using a Poisson process with three different rates: \( 1.4 \lambda_N \), \( 5 \lambda_N \), and \( 20 \lambda_N \) sessions/minute. These values, coupled with the distribution of Skype call duration, leads to 10%, 30%, and 98% of nodes simultaneously involved in a media session, respectively. Simulation lasts enough to exit from transient period; presented results are referred to the steady state.

B. Overlay topology evaluation

This simulation, which aims at demonstrating that our protocol really creates a scale-free network, considers the two main parameters that characterize the overlay topology: the clustering coefficient and the in-degree of nodes [7]. The clustering coefficient is an index of the presence of clusters (i.e., strongly connected components) and is computed for each node as the number of links between the vertexes within the neighborhood divided by the number of links that could possibly exist between them: the greater is the clustering coefficient the greater is the edge number between nodes of the neighborhood. The first requirement the overlay must meet to be a scale-free network is to have an average clustering coefficient higher than the one of a random graph. Figure 1 shows this comparison, proving that the average clustering coefficient of the DISCOS overlay is always higher. In details, the average clustering coefficient of the distributed connectivity service decreases when the network size grows as expected, asymptotically converging to a value that is about 20 times the clustering coefficient of a random graph. We also verified that, at all network sizes experimented, the coefficient remains almost constant in time. Concerning the in-degree, the requirement to meet is that the distribution of node degree must follow a power-law \( P(k) = ck^{-\gamma} \), where \( P(k) \) is the probability that a node has \( k \) connections and \( c \) is a normalization factor. Figure 2 shows how the distribution of in-degree values obtained through simulation results to follow a power law. In particular, the distribution well fits a power law with \( c = 0.7 \) and \( \gamma = 1.5 \). These tests validate our overlay construction model, showing the resulting topology really evolves in a scale-free network.

In order to prove the effectiveness of the DISCOS topology, we compare our solution with a distributed system where the information is randomly spread and nodes to query during lookup procedures are randomly chosen among peers in the cache. Figure 3 depicts the average number of peers that have to be contacted to reach an available SIP relay for both DISCOS and the randomized overlay. While the advertisement rate and the TTL value remain the same, the figure shows that in DISCOS the number of peers contacted is sensibly lower. Furthermore, the ratio between the lookup performances offered by the two policies grows with the network size, reaching a value of about 5 with 30000 users. This proves that searches towards hubs discussed in [8] are efficient also when an overlay is used to lookup a single resource provided by many nodes.

These tests prove the effectiveness and the scalability of DISCOS. In particular, results show how the scale-free topology ensures overlay efficiency with a limited message rate (each peer sends an advertisement message every 60 minutes) with a small TTL (equal to 2) and a limited cache size (10 entries): this results in a reduced per-node-overhead, confirmed by the fact that 99% of nodes process less than 7 advertisement messages per minute and remaining 1% process a number of messages that varies between 8 and 48 messages per minute. This confirms that hubs, which sustain a heavier load, should be chosen carefully according to their
computational and bandwidth resources, e.g., using the dynamic protocol proposed by Chawathe et al. for the Gia P2P network [9].

![Figure 1: average clustering coefficient evaluation](image1)

![Figure 2: in-degree power-law distribution](image2)

**C. Media sessions relaying performance**

This section aims at analyzing the overlay support for media sessions, in particular when hole punching fails and relaying is needed. To prevent resource wasting, a media relay is typically acquired by a UA just before the establishment of a media session. Various types of media flows are considered, differing in the amount of consumed bandwidth. In particular, assuming $b$ bit/s is the consumed bandwidth unit, five types of flows requiring $nb$ ($1 \leq n \leq 5$) bit/s are defined. The flow type is randomly selected (with uniform distribution) when a session occurs. We also define $B_i$ as the amount of bandwidth that peer $i$ can offer for relaying media sessions. For the sake of simplicity, $B_i$ is assumed to be the same for each connectivity peer and equal to $5b$ bit/s. However, in a real scenario this value could vary according to node capabilities.

We start the evaluation of the DISCOS support for media sessions from the estimation of the failure probability as it is the parameter that mainly affects the quality of service perceived by users. A session can fail for two reasons: (i) no available relay is found, or (ii) the relay is found but suddenly becomes unavailable during the session (e.g., because it disconnects from the network). With respect to the first problem, we never observed such an event during simulation: a UA with limited connectivity has always found a media relay. This result suggests that, with the given assumption about the number of media sessions needing a relay, the probability for this event to occur in a DISCOS environment can be considered negligible. The second issue could be mitigated by implementing proper relay backup policies. As shown in Figure 4, the media session can fail in about 0.6-0.65% of cases, but the selection of one backup relay (that takes care of the communication in case the first relay fails) sensibly reduces this probability, and further reductions are possible increasing the number of relay nodes. The blocking probability remains low even in the unlikely case in which 98% of the users are involved in a call. The overhead deriving from the search of backup relays is depicted in Figure 5, which plots the average number of peers that have to be contacted to find $K$ available media relays. For a reasonable number of simultaneous sessions, this value remains low. However, we set the number of backup relay nodes to 1, which is a good trade-off between the achieved session failure probability and the additional complexity that results when a UA has to search a backup relay node before starting media sessions.

![Figure 3: average number of contacted peers to find a SIP relay](image3)

Finally, we analyzed the distribution of load among connectivity peers. In particular, Figure 6 shows the distribution of the fraction of the available bandwidth $B_i$ effectively consumed in each peer. Given the assumed value for $B_i$ (equal to $5b$ bit/s for each peer) and the distribution of the amount of bandwidth consumed by media flows (assumed to be uniform between $b$ and $5b$ bit/s), the results showed in the figure lead to conclude that sessions are reasonably uniformly distributed across the infrastructure. Thus, a good load balancing among peers is guaranteed.
confirm that the overhead for managing the overlay is low, that each host is able to locate a suitable connectivity peer with a limited number of messages (hence, in a very short time), and the blocking probability of a new media call is negligible even for very high load. Although our simulations cannot simulate a nationwide network (for processing/memory problems), we are confident that results can be extended to such an environment, because the distributed infrastructure is based on the scale-free topology, which is the key to achieve these results ensuring overlay scalability and robustness.

Future work aims at validating the proposed infrastructure in non-SIP environments (e.g., creating a self-organizing, ad-hoc infrastructure) and taking care of security issues, e.g. to prevent untrusted (and malicious) peers from being able to compromise the overlay topology.

V. ACKNOWLEDGMENT

The authors would like to thank Marco Mellia was instrumental to obtain a proper characterization of Skype user agents.

VI. REFERENCES