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
{luigi.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, middleboxes such as 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 application 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 (NATs)*, which enable the reuse of private addresses, and/or *firewalls*, which are used to secure corporate networks and internal resources. 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 the applications that use different paradigms, such as VoIP and P2P. Particularly, middleboxes prevent nodes behind them to be directly contacted from external nodes. For example, an internal host might not have problems to start a data transfer to an external host but the vice versa (e.g, an incoming VoIP call) may be impossible. Thus, proper strategies for middlebox traversal are required in order to enable the seamless communication between hosts, no matter where they are located. Among the known strategies, *hole punching* and *relaying* [1] represent the ones more frequently used. The common idea is to make the middlebox believe that the internal host begins the communication. The middlebox will then create a temporary channel with the remote host, thus allowing the delivery of packets coming from the outside world. Particularly, the *hole punching* forces each internal host to keep a persistent connection with an external rendez-vous server located on the public Internet. This creates a sort of "hole" that can be used by an external host to contact the internal host directly. If hole punching fails, for example if hosts are behind symmetric NATs, the *relaying* represents the last chance: internal hosts maintain 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 advertises the IP address of the relay server as one of its addresses, 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], which are among the applications that suffers most from middleboxes limitations. Two solutions have been defined in this context. SIP messages directed to the destination User Agent (UA) are delivered with a relay-based approach, which exploits an intermediate public SIP proxy [3]. For media flows, the Interactivity Connectivity Establishment (ICE) [4] protocol has been proposed. ICE is an integrated solution defined to discover NAT bindings and to execute the hole punching for media streams. In addition, ICE supports also media relaying based on the TURN [5] protocol. Both the hole punching mechanism of ICE and TURN rely on STUN [6], a client-server protocol consisting in two messages, *Binding Request* and *Binding Response*. These messages are enough for implementing the hole punching procedure [1], while TURN needs to extend the STUN protocol to establish communication channels with relays, called *TURN servers*. STUN can also be used to implement a middlebox behavior discovery service [7], which can be used by internal hosts to determine the type of NAT/firewall they are behind to.

The presented middlebox traversal solutions rely on centralized servers that provide rendez-vous and relay capabilities. However, 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, in which servers may be required to handle a huge amount of traffic (both SIP signaling messages and media datagrams), thus requiring a high amount of computational resources and bandwidth. The server acting as relay for SIP (i.e., the SIP proxy) has also to handle the traffic generated by keep-alive messages that UAs behind middlebox periodically send to it. Keep-alive messages are necessary to

maintain the communication channel with the server and thus to guarantee these UAs can be always reached. This could result in a high overhead. For example, according to the NAT binding timeout reported in [3], in a SIP domain including 1.5 millions of UAs with limited connectivity, the central server has to handle about 50000 keep-alive messages per second.

This paper proposes a distributed architecture — referred to as *DIStributed COnnectivity 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 the above described rendez-vous and relay functionalities in the UAs. Each globally reachable UA with enough resources can provide such services to UAs with limited connectivity. A major emphasis is given to the overlay design, as it is a key point for ensuring a fast "service lookup" (i.e., to find a peer that still have enough resources for offering the connectivity service), which is instrumental for providing an adequate quality of service to the users. In particular, we show how a scale-free topology can fit this requirement and we propose an overlay construction model that can be used to build such topology.

DISCOS is somewhat orthogonal to P2PSIP [8], although both being based on P2P technologies. In fact, P2PSIP is mainly a solution for distributed lookup whereas DISCOS offers a solution for middlebox traversal.

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 that cannot be studied and evaluated by third parties, therefore limiting the ability to understand exactly how these problems are solved. For example, in the Skype analysis presented in [9] and [10] the authors could give only partial explanations about its NAT and firewall traversal mechanisms. Their experiments pointed out that nodes with enough resources can become *supernodes* and provide support for NAT and firewall traversal. In particular, they offer relay functionalities and probably run a sort of STUN server that other nodes use to discover the presence (and to determine the type) of NAT and firewall in front of them. Therefore, it is clear that a node behind NAT has to connect to a supernode in order to be part of the Skype network, but no information could be given about the supernode discovery and selection policies. Also supernode overlay topology is almost completely unknown. Thus, there is no way to evaluate the effectiveness of these solutions. On the other hand, here we propose a distributed architecture for middlebox traversal whose scalability and robustness are discussed and evaluated. In addition, the solution 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.

II. Operating Principles

A. Distributed Connectivity Service

The *Distributed Connectivity Service (DISCOS)* extends current centralized NAT and firewall traversal solutions by distributing rendez-vous and relay functionalities among UAs. Relaying and hole punching service for media flows is implemented by integrating a STUN/TURN server in each UA. The TURN server is also used to support relaying of SIP messages. However, DISCOS can be easily modified to offer relaying of SIP messages by integrating SIP proxy functionalities in each UA, leading to a distributed implementation of [3].

UAs with enough resources (e.g. a public network address, a wideband Internet connection and free CPU cycles) become what we define a *connectivity peer* and start offering connectivity service. In particular, connectivity peers can act as both SIP relay (leveraged by UAs with limited connectivity for receiving SIP messages) and media relay. Connectivity peers can also offer support to the hole punching procedure for media session establishment, thus operating as a distributed rendez-vous server. In addition, connectivity peers also provide support for middlebox behavior discovery [7]. UAs with limited connectivity can locate and attach to an available peer whenever they need one of these services.

Connectivity peers are organized in a P2P overlay and their knowledge is spread through proper advertisement messages, thus building an *unstructured gossip-based network*. Structured networks, characterized by additional overhead due to the maintenance of the structure, are not considered because their excellent lookup properties are not required. In fact, DISCOS uses the overlay only to find the first available connectivity peer and not for locating a precise resource.

It is worth noticing that, since DISCOS distributes existing middlebox traversal functionalities among peers, it is also totally compatible with current middleboxes and their traversal solutions. This enables a smooth deployment of the proposed solution.

B. Overlay Topology

In order to enable DISCOS to locate an available peer for 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 to find 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 average path length between nodes, the higher is the depth of the flooding (hence the load on the network) that is needed for an adequate spread of the information. 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. In fact, nodes maintain a cache that

should be kept small in order to reduce the overhead required to manage all the entries. This limits the number of peers known at each instant. 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 different 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 itself. 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 [11], 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 having that node in their cache. Thus, nodes can be considered hubs when they are in the cache of several peers, i.e., when they are highly popular. Hubs frequently receive advertisement messages from a large set of different nodes, so they frequently update their cache. In particular, if advertisement 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. [12] 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 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 at the expense of a non-uniform distribution of the number of messages handled by nodes: the higher is the popularity of a node, the larger is the number of advertisement messages received. 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 [11] 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 make *M* peers available to other nodes through a Bootstrap Service. When a node joins the overlay for the first time, it queries the Bootstrap Service 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 provided by the Bootstrap Service 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, the popularity is computed autonomously by each node through a simple approximated metric based on the number of received advertisement messages that contain such node. In our approximated model, preferential attachment is implemented by forcing 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 nodes to insert highly popular peers (hubs) in their cache, thus building and maintaining the scale-free topology. In summary, new nodes use the peers known through the Bootstrap Service as "bootstrap" nodes, then they learn most popular ones through the received advertisement messages and start to perform preferential attachment. Furthermore, incoming nodes that already know peers discovered during their previous visits can avoid bootstrap procedure by attaching directly to them. The presented topology is shown in Figure 1.

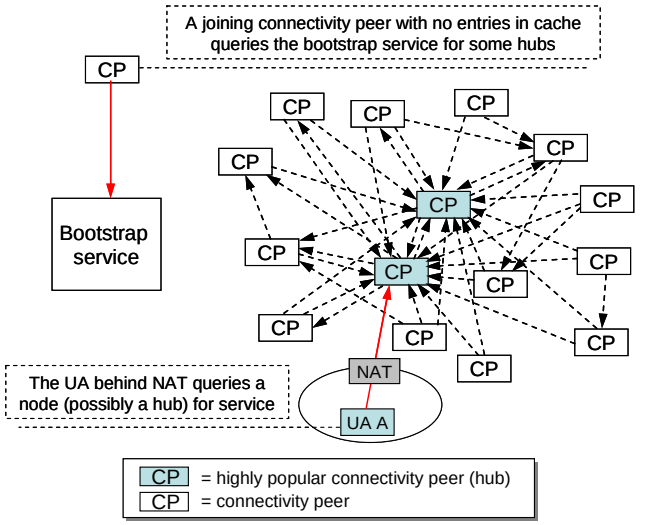

Figure 1. DISCOS overlay topology

It is worth noticing that different Bootstrap Services can be used to create disjoint overlays as joining peers that fetch nodes from different Bootstrap Services will start exchanging advertisement messages with different connectivity peers. This enables the possibility to deploy different DISCOS overlays in different geographical areas of a SIP domain. If a location-aware Bootstrap Service selection policy is adopted, users can find a connectivity peer that is close to them, thus preserving the user-relay latency achieved by current centralized solutions, where different servers can be used at

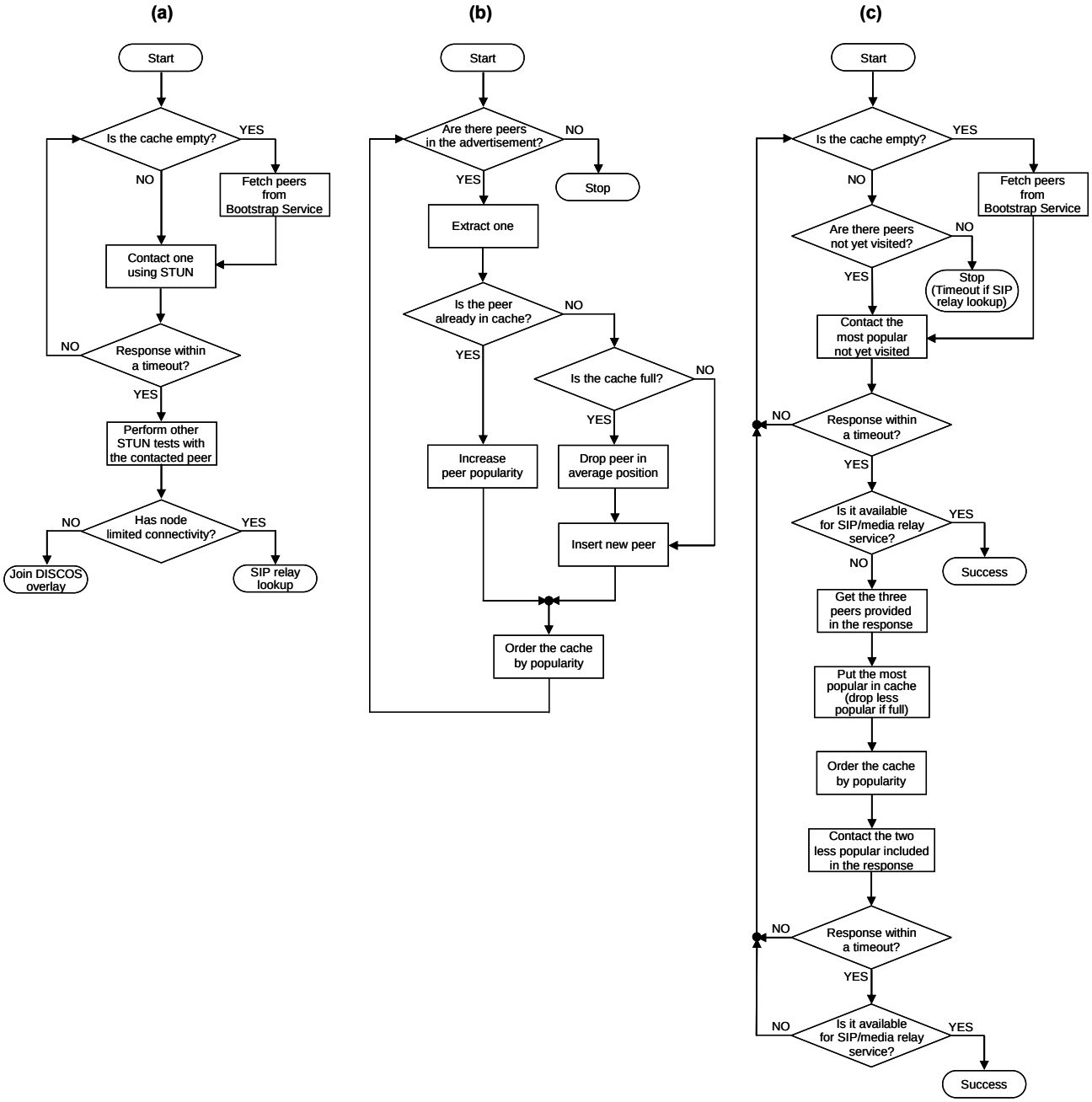

Figure 2. Operation of DISCOS when (a) a node joins the SIP domain, (b) a node in the overlay receives an advertisement message, and (c) a node performs a SIP/media relay lookup

different locations.

The implementation of the Bootstrap Service is highly customizable. A possible solution consists in deploying *M* static peers and preconfiguring their addresses on each UA. A more flexible approach (considered in the following) consists in deploying multiple bootstrap servers reachable through appropriate DNS SRV entries configured in the DNS. Each bootstrap server stores information about *M* connectivity peers that spontaneously register themselves when they join the overlay. Multiple bootstrap servers are deployed for redundancy and load balancing purposes. Proper DNS configuration can enable a location-aware Bootstrap Service selection.

C. Protocol Overview

Whenever a UA joins the SIP domain, it has to determine if it can become a connectivity peer or it is behind a middlebox. This is done by contacting a connectivity peer and exploiting its STUN functionalities [9]. The described bootstrap procedure is performed if it does not know any active peer. The flow chart related to the join procedure is shown in Figure 2(a).

If the UA can become a connectivity peer, it checks the number of addresses registered on each bootstrap server and, if it is smaller than a fixed bound *M*, it adds itself to the list. Then, it sends an advertisement message to the known peers in order to announce itself. The UA is now part of the

DISCOS overlay and it starts receiving messages from other nodes, thus gradually filling its cache with new peers. A proper peer advertisement policy is adopted in order to implement preferential attachment (thus building and maintaining the scale-free topology) and to enable caches to be refreshed with lightly loaded peers (thus having potential nodes available for the service). In particular, advertisement messages include the sender node, the two most popular peers it knows (enabling preferential attachment), and the two less popular peers it knows (spreading the knowledge of 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 a positive value, 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. 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 big hubs and lowly popular peers. Figure 2(b) details the operations of a peer when receives an advertisement message.

UAs with limited connectivity have a different behavior, since they essentially exploit DISCOS features to find SIP relays (they choose a connectivity peer as relay for SIP messages as soon as they join the SIP domain; in addition, they select another when the current one disappears) and media relays (when they need one to establish a media session). A UA with limited connectivity performs these lookups by contacting the most popular peers in its list, which can accept or decline 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 searches because it is probably 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 cannot be established. Relay lookup procedure is shown in Figure 2(c).

UAs with limited connectivity also receive ad-hoc messages from their relays containing three highly popular peers, which allow them 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. Broken hubs (e.g., because of a network failure) are detected through a timeout: if a hub does not reply to a query, the UA can query one of the others hubs in its cache. If no peers are available, the UA fetches again the registered ones from the bootstrap server; however, this situation is unlikely to occur as UAs with limited connectivity periodically receive new hubs from their SIP relays.

This protocol could be integrated in SIP as well as implemented separately. The former approach is more straightforward as it simply consists in defining new SIP header fields. The latter one is more efficient, especially concerning the message size. In fact, the human readable nature of SIP messages would result in advertisement messages large about 800 bytes.

D. Security issues

The deployment of a P2P architecture for providing connectivity service raises several security issues that are different than in centralized solutions. In DISCOS, like in many other distributed systems, the control of the consequences of malicious behavior of nodes can be more difficult than in the centralized counterpart. Much effort has been expended during the last past years in investigating these issues in the context of P2PSIP overlays [8][13][14], which have to deal with similar concerns as they replace centralized SIP proxies for user location. Some solutions have been proposed and can be seamlessly applied in DISCOS. For example, in [14] public key certificates are distributed among users to allow them to verify the origin and the integrity of messages. Analogously, certificates can be used in DISCOS to authenticate advertisement messages, so that they can be considered trusted. This limits the operation of malicious peers as they can be easily traceable. This and other P2PSIP derived security policies certainly need further improvements in order to better fit specific DISCOS requirements. However, we are confident that effective results can be obtained with minimal modifications because, as mentioned above, security issues to address are similar in the two environments. This additional effort is left for future work.

III. OVERLAY SIMULATION

A. Simulations background

We developed a custom, event-driven simulator to evaluate the effectiveness of the proposed solution. In particular we were interested in proving its scalability and validating its algorithms. Thus, we implemented a simulator supporting the following four operations: node arrival/departure, media session setup/teardown, SIP relay lookup (triggered when a node with limited connectivity joins the network or when its current SIP relay disappears), and media relay lookup (that occurs when a node need a relay to perform a media session).

Simulations are referred to a single SIP domain. Node arrivals and call occurrences are modeled using a Poisson process, while node lifetime and call length are extracted from real Skype traffic coming from/to the network of the University campus in order to approximate the behavior of

real VoIP networks. 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 (some lasting several days on a Dual Xeon 3 GHz processor) and significance of results. In order to test our solution within different traffic load scenarios, three different rates are used for media session occurrences: $1.4\lambda_N$, $5\lambda_N$, 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.

Statistics presented in [15] 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 nodes have limited connectivity with probability $P_{LC} = 0.74$ and media sessions directed to these nodes need relaying with probability $P_{MR} = 0.18$. Whenever a node joins the SIP domain, two different actions can be performed at simulation level: if it is tagged as a node with limited connectivity (with probability P_{LC}), it triggers a SIP relay lookup, otherwise it will join the DISCOS overlay as a connectivity peer. Media sessions are possible between each pair of nodes (selected randomly). When a node behind NAT is contacted, a media relay lookup is triggered by this node with probability P_{MR}.

The number of UAs with limited connectivity to which a peer can simultanously provide SIP relay service is set to 10, advertisement messages have a TTL equal to 2, their sending interval is set to 60 minutes, and peer's cache are supposed to contain 10 entries. Furthermore, the number of peers registered in the bootstrap server (which is supposed to be unique and reachable by nodes) is set to 20. Simulation lasts enough to exit from transient period; presented results are referred to the steady state.

B. Overlay topology evaluation

First simulation aims at demonstrating that our protocol creates a scale-free network among connectivity peers. In particular, we consider the clustering coefficient and the in-degree of nodes [11]. The clustering coefficient of a node is defined as the number of links between its neighbor nodes divided by the number of links that could possibly exist between them. In order to be a scale-free, an overlay must have an average clustering coefficient higher than the one of a random graph obtained in the same conditions, which is clearly proved in Figure 3(a). In details, the average clustering coefficient for DISCOS decreases when the network size grows, 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 follows 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 3(b) shows that the distribution of in-degree values obtained through simulation 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(c) 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 performances obtained by the two policies increases with the network size, thus demonstrating the scalability properties of our solution.

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). We have also evaluated the number of advertisement messages that connectivity peers have to handle in our simulated SIP domain including 30000 UAs: 99% of nodes processes less than 7 advertisement messages per minute and the remaining 1% processes a number of messages that varies between 8 and 48 messages per minute, thus resulting in a reduced per-node overhead. However, this confirms that hubs should be chosen carefully, preferring nodes with enough computational and bandwidth resources, e.g., using the dynamic protocol proposed by Chawathe et al. for the Gia P2P network [16].

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 chosen by a UA right 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 $(1 \le n \le 5)$ bit/s are defined. The flow type is randomly selected (with uniform distribution) when a new session starts. 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

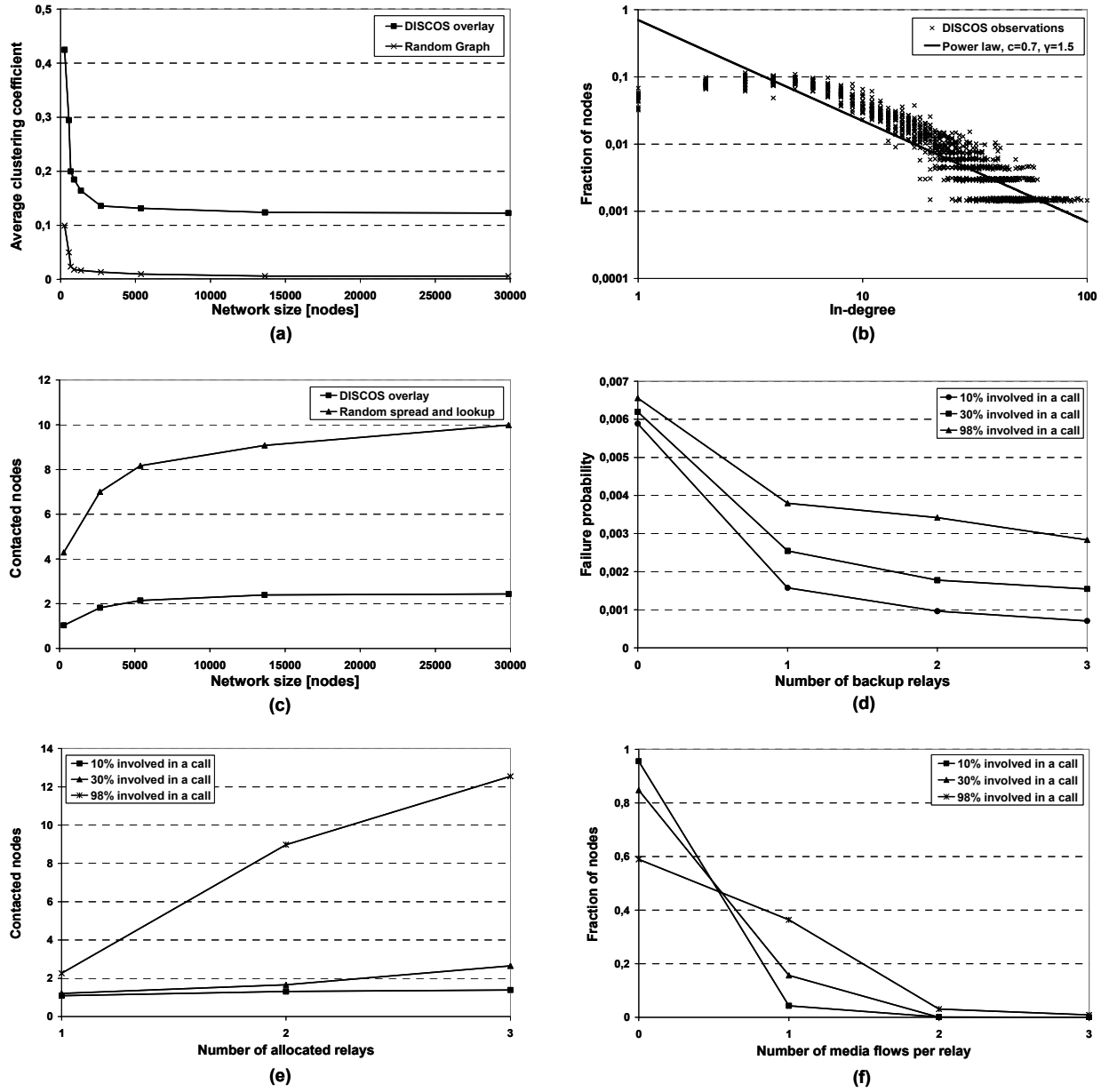

Figure 3. Simulations results: (a) average clustering coefficient evaluation, (b) in-degree power-law distribution, (c) average number of contacted peers to find a SIP relay, (d) media session failure probability versus number of allocated backup relays, (e) average number of peers contacted to allocate K relays, and (f) bandwidth consumption distribution

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 was always able to find a media relay. This result suggests that, with our assumptions about the number of media sessions requiring 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 3(d), the media session can fail in about 0.6-0.65% of cases, but the selection of a single 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 (i.e., almost all users are at the phone). The overhead deriving from the search of backup relays is depicted in Figure 3(e), 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 reasonable trade-off between the probability of a session drop and the additional complexity that results when a UA has to search a backup relay node before starting media sessions.

Finally, we analyzed the distribution of load among

connectivity peers. In particular, Figure 3(f) shows the distribution of the number of media flows simultaneously handled by media relays. It can be observed that, although media flows have different bandwidth requirements, the great part of relays simultaneously handles no more than one media session. Thus, a good load balancing among peers is guaranteed.

IV. Conclusions

This paper presents a distributed infrastructure, called DISCOS, which aims at providing connectivity service to hosts behind middleboxes. This solution extends current centralized approaches (and overcomes their scalability and robustness limitations) by integrating middlebox traversal functionalities into edge nodes. The paper also presents the mechanisms that can be used to manage such infrastructure and exploit its services. The proposed infrastructure is based on an unstructured peer-to-peer paradigm and has been proved to be extremely effective in locating suitable relays and distributing media sessions evenly among the available connectivity peers. Results confirm that the overhead for managing the overlay is low, that each host is able to locate a suitable connectivity peer with a small 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 and more exhaustively address security issues.

V. Acknowledgment

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

VI. References

[1] Bryan Ford, Pyda Srisuresh, Dan Kegel, “Peer-to-Peer Communication Across Network Address Translators,” *USENIX Annual Tech. Conf.*, Anaheim, CA, Apr. 2005.

[2] J. Rosemberg et al., “SIP: Session Initiation Protocol,” *IETF Std. RFC 3261*, June 2002.

[3] C. Jennings, Ed., R. Mahy, Ed., “Managing Client Initiated Connections in SIP,” http://tools.ietf.org/html/draft-ietf-sip-outbound-11, Nov. 2007.

[4] J. Rosenberg, “Interactive Connectivity Establishment (ICE): A Protocol for NAT Traversal for Offer/Answer Protocols,” http://tools.ietf.org/html/draft-ietf-mmusic-ice-18, Mar. 2008.

[5] J. Rosenberg, R. Mahy, P. Matthews, “Traversal Using Realys Around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN),” http://www3.tools.ietf.org/html/draft-ietf-behave-turn-07, Feb. 2008.

[6] J. Rosenberg, R. Mahy, P.Matthews, D. Wing, “Session Traversal Utilites for (NAT) (STUN),” http://tools.ietf.org/html/draft-ietf-behave-rfc3489bis-15, Feb. 2008.

[7] D. MacDonald, B. Lowekamp, “NAT Behavior Discovery Using STUN,” http://www3.tools.ietf.org/html/draft-ietf-behave-nat-behavior-discovery-03, Feb. 2008.

[8] D. A. Bryan, B. B. Lowekamp, “Decentralizing SIP,” *ACM Queue*, vol. 5, no. 2, March 2007.

[9] S. A. Baset, and H. Schulzrinne, “An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol,” *IEEE Int. Conf. on Computer Communications (INFOCOM 2006)*, Barcelona, Spain, Apr. 2006.

[10] P. Biondi, F. Desclaux, “Silver Needle in the Skype,” *Black Hat Europe 2006*, Amsterdam, the Netherlands, Mar. 2006.

[11] R. Albert, A.-L. Barabási, “Statistical mechanics of complex networks,” *Review of Modern Physics*, 74, 47-97, 2002.

[12] L. A. Adamic, R. Lukose, A. R. Puniyani, B. A. Huberman, “Search in Power-law Networks,” *Physical Review*, E 64, 2001.

[13] J. Seedorf, “Security Challenges for Peer-to-Peer SIP,”*IEEE Network*, vol. 20, no. 5, Sept. 2006.

[14] C. Jennings, B. Lowekamp, E. Rescorla, S. Baset, H. Schulzrinne, “Resource Location And Discovery (RELOAD),” http://www.p2psip.org/drafts/draft-bryan-p2psip-reload-04.txt, June 2008.

[15] Martin Casado and Michael J. Freedman, “Peering through the Shroud: The Effect of Edge Opacity on IP-based Client Identification,” *USENIX/ACM Int. Symp. on Networked Systems Design and Implementation (NSDI 2007)*, Cambridge, MA, April 2007.

[16] Y. Chawathe, S. Ratnasamy, L. Breslau, Nick Lanham S. Shenker, “Making Gnutella-like P2P Systems Scalable,” *ACM Int. Conf. of the Special Interest Group on Data Communication (SIGCOMM 2003)*, Karlsruhe, Germany, Aug. 2003.