Framework for Decentralizing Legacy Applications

Jani Hautakorpi
and Gonzalo Camarillo
Ericsson Research Nomadiclab
FI-02420 Jorvas, Finland
Email: firstname.lastname@ericsson.com

David López
Universidad de León
E-24071 León, Spain
Email: david.lopez@unileon.es

I. INTRODUCTION

Peer-to-peer (P2P) networks and applications are a trend in the telecommunications landscape. When we talk about P2P networks in this paper, we refer to networks which comprise multiple nodes running some kind of data distribution mechanism, such as Distributed Hash Table (DHT) algorithm. Currently there is a considerable number of Internet users using P2P applications for interpersonal communication (e.g., Skype) and for file sharing (e.g., BitTorrent).

Some of the P2P applications have replaced, to a certain extent, their centralized counterparts. An example from this is file sharing. Several decentralized P2P applications (e.g., BitTorrent, eDonkey, and Gnutella) are currently used for very similar use cases as centralized File Transfer Protocol (FTP) was, and is, used for. That being said, there are still legacy applications that use traditional, centralized infrastructure rather than decentralized approach.

In this paper we present a generic framework for decentralizing legacy applications. Our decentralization framework makes it possible to use legacy applications without relying on any centralized infrastructure, including the Domain Name System (DNS). We present this decentralization framework as an alternative to the centralized infrastructure and leave the comparison between the two outside the scope of this paper and for future work. One natural high-level benefit of the decentralization framework is that there is no longer need to maintain centralized infrastructure.

We had three specific design goals for our framework. The first goal was to make the framework as generic as possible, so that it would enable the decentralization of multiple legacy applications. Even though we have focused on three specific applications, VoIP, email, and Web, we believe that our framework is suitable also for other applications using centralized servers mainly as a means to provide a rendezvous service.

The second goal was to preserve legacy applications as they are today. In other words, we do not want to change legacy applications at all, but rather adapt P2P networks for them.

The third goal was to allow the sharing of a data distribution mechanism (e.g., DHT algorithm) among several applications. This is especially important in environments where nodes are behind Network Address Translators (NATs). One of the main problems NAT devices create in P2P networks is that they require frequent keep-alive messages for each connection. This problem is emphasized on wireless, battery powered devices.

We have implemented a proof-of-concept prototype of our decentralization framework. The main purpose of the prototype is to validate that the design of our decentralization framework is correct and that it fulfills its design goals.

II. BACKGROUND AND RELATED WORK

A common way to enable the decentralizing of applications and services is to use some data distribution mechanism, as we use in our decentralization framework. One option as a data distribution mechanism is a DHT algorithm, such as Chord [4] or Pastry [5]. Different DHTs differ on their geometries, robustness, scalability properties and such, but all of them provide one main operation: when given a key, they locate a node responsible for corresponding object (object can be, e.g., a service or resource).

There are some existing efforts to decentralized legacy applications that relate to our decentralization framework. In the following sections we present these efforts.

A. P2PSIP

The technology called P2PSIP combines SIP [6] and P2P networks in a novel manner. Its main purpose is to decentralize the location service of SIP and to provide the ability for interpersonal communications. Users of a P2PSIP network, as the users of a SIP network, are identified by using SIP URIs (e.g., sip:alice@example.org). Two of the earliest research papers on P2PSIP were [1] and [7], and a number of papers have been written since, for example [8]. The P2PSIP technology is currently being standardized at the Internet Engineering Task Force (IETF). The IETF has produced a set of documents that, for example, describe the main concepts of P2PSIP [9] and the
peer protocol, Resource Location and Discovery (RELOAD) [10].

RELOAD is a protocol that is used between peers in a P2PSIP network. It is noteworthy that RELOAD, as it is today, is a result of merging multiple peer protocol proposals together, and one of them was Peer-to-Peer Protocol (P2PP) [11]. RELOAD provides a cooperative storage and messaging service between a set of peers forming an overlay. It requires an implementation of a data distribution mechanism, and the currently specified data distribution mechanism is a variant of Chord [4] algorithm. The suitability of a set of selected DHTs for P2PSIP has been evaluated in [12]. A notable characteristic of RELOAD is that even though it is designed for P2PSIP, it can also be used by other applications.

The security of a P2PSIP network can be based either on self-signed certificates or on centralized Certificate Authority (CA) distributing certificates to users. Security model based on self-signed certificates is easier to deploy but it is less secure than the one based on a centralized CA.

The P2PSIP application that a user is using in a P2PSIP network can be either a specifically implemented P2PSIP application, or it can be a traditional SIP User Agent (UA). When traditional SIP UAs are used, then there is a need to use a specialized, P2PSIP-aware outbound proxy, such as SIPpeer [13].

B. ePOST
ePOST [2], which is based on POST [14], provides a decentralized email service. The users of ePOST can use the system with legacy email applications (such as Mozilla Thunderbird, Microsoft Outlook, and Apple Mail). Users of ePOST, as the users of traditional email system, are identified by email addresses (e.g., mailto:bob@example.org).

POST is a layer built on top of Pastry [5] and it provides a decentralized message storage, single-writer logs, and event notification. ePOST uses the decentralized message storage to store email messages, single-writer logs to present the state of mailboxes, and event notification for notifying the users - or rather their local proxies - when a new email arrives.

The underlying assumption in ePOST is that the system has a centralized CA handing out certificates to the users of ePOST. The CA provides a foundation for security features of ePOST (e.g., Sybil attack [15] prevention).

Currently ePOST is implemented as a local proxy which support Simple Mail Transfer Protocol (SMTP) [16], Internet Message Access Protocol (IMAP) [17], and Post Office Protocol version 3 (POP3) [18]. The local proxy also participates to a Pastry overlay and contributes local resources to the overlay.

C. DDNS and Web-over-SFR
There are efforts which have used DHT algorithms to either provide a similar service as DNS, DDNS [3], or to provide Semantic-Free Referencing (SFR) service for Web, Web-over-SFR [19]. The equivalent of DNS queries are done based on plain hostnames in DDNS (e.g., example.org) and based on whole Uniform Resource Locators (URLs) in Web-over-SFR (e.g., example.org/index.html). Both of these approaches use DHash [20] for storing bindings between references and IP addresses.

DHash is a block-storage layer that provides load balancing, caching, and replication for applications using it. These features are provided transparently to the applications. Chord, in turn, provides a distributed lookup system for DHash.

Both, DDNS and Web-over-SFR, make it possible to sign the data on the overlay network by using public key cryptography. Signing is used, so that the requestor of the data can validate it and can be assured of its origin.

The DDNS is implemented as a loopback server listening on the localhost. The loopback server accepts DNS queries and preforms corresponding Chord overlay lookups. There are two different implementations of Web-over-SFR. The first one is a SFR client library that is linked to the applications. The SFR client library can resolve the references, that it gets from the applications, by making P2P lookups in a Chord network. The second implementation is a Hypertext Transfer Protocol (HTTP) [21] proxy that translates traditional URLs to references which are then resolved by using a SFR client library.

III. DECENTRALIZATION FRAMEWORK
Our decentralization framework for legacy applications is designed to be as generic as possible, support legacy applications as they are today, and enable the sharing of a data distribution mechanism. It is noteworthy that the actual applications users are using (e.g., Mozilla Thunderbird) are not decentralized, but the servers (e.g., SMTP servers) providing the service are. The decentralization framework has two main components: the Local Extendable P2P Proxy (LEPP) and the augmented DNS resolver, see Fig. 1.

The LEPP component contains a P2P daemon and protocol-specific modules. The P2P daemon is an entity that contains an implementation of a data distribution mechanisms and a peer protocol. We have chosen a variant of Chord DHT [4] as our data distribution mechanism and P2PP [11] as our peer protocol, even though the framework itself also could host other data distribution mechanisms (e.g., Pastry [5]) and peer protocols (e.g., RELOAD [10]). From here on we talk about P2PP, as opposed to data distribution mechanism, so that the framework would be more concrete and easier for the reader to understand.

The protocol-specific modules are behaving as proxies and each of them contains an implementation of an application layer protocol (e.g., SIP). Even though the Fig. 1 depicts only three modules, VoIP, email, and web module, the LEPP is extendable and could host as many modules as necessary. Each module operate in a way that the egress submodules (e in the figure) write a mapping between a name (e.g., SIP URI, email address or hostname) and an IP address of a node to the overlay, and the ingress submodule (i in the figure) reads the IP address of the recipient from the overlay. In other words, the modules provide a rendezvous service. This is somewhat similar approach as in Internet Indirection Infrastructure (i3).
[22]. However, a significant difference between our approach and i3 is that the actual communication between users is done directly in our approach and indirectly via intermediaries in i3.

One of the design goals for the decentralization framework was to support the sharing of a data distribution mechanism. This is achieved with an arrangement where all the modules in LEPP use the same P2P daemon. In our prototype we have chosen to use Extensible Markup Language Remote Procedure Call (XML-RPC)\(^1\) as the interface between the modules and the P2P daemon.

The second main component of our framework is the augmented DNS resolver. The additional functionality that our decentralization framework requires from the DNS resolver is a simple one; when an application is requesting an IP address of a domain name which has "p2p" as the Top-Level Domain (TLD) (i.e., the domain name ends with ".p2p"), the augmented resolver returns the address of the localhost (e.g., 127.0.0.1) in the DNS response. Our approach borrows an idea of appending text to the URLs from CoralCDN\(^2\). However, in our approach the DNS queries, the ones ending with ".p2p", are processed on a local node rather than on dedicated DNS servers as in CoralCDN.

There are several options on how the above described functionality can be added to the DNS resolver. The first option is to intercept and modify the system calls between applications and the DNS resolver. The second option is to set the DNS server address of the operating system to the address of the localhost and then set up a local DNS relay for handling the DNS messages (DDNS\(^3\) uses this approach). The third option is to modify the actual DNS resolver itself. In our prototype we have chosen the first option.

A. VoIP Module

The VoIP module in the LEPP enables the decentralization of VoIP applications (such as X-Lite\(^2\)). This paper focuses only on VoIP applications that use SIP as a signaling protocol. Our VoIP module is somewhat similar to P2PSIP (see Section II-A), and resembles especially the adaptor functionality of SIPpeer [13]. In SIPpeer the peer protocol, which was SIP, and the data distribution mechanism was a part of the SIPpeer itself. In contrast, our framework uses a separate P2P daemon which is shared with other decentralized applications. Another difference is that SIPpeer required a user to manually input the address of the SIP outbound proxy to his/her VoIP application, but in our framework the user does not have to input the address of the outbound proxy, just his/her own SIP URI. The reason is that in our framework the augmented DNS takes automatically care of directing originating SIP requests to the local SIP proxy, which is LEPP.

When the user starts the VoIP application, it first registers itself to a P2P network so that other users can reach it, see Fig. 2. First the legacy VoIP application does a DNS query (1) which contains the domain name part of user’s Address-of-Record (AOR). The augmented DNS resolver receives the DNS query and inspects if the TLD of the domain name is "p2p". If it is, the augmented DNS resolver returns (2) the address of the localhost, otherwise it will make an external query (3) which will result in a DNS response (4) containing the IP address of the localhost. The VoIP module then uses this IP address to connect to the local SIP proxy, which is LEPP.

Fig. 2. Sequence diagram of a decentralized VoIP application

\[^{1}\text{See, http://www.xmlrpc.com}\]

\[^{2}\text{See, http://www.counterpath.com}\]
DNS query and return its answer.

Now the VoIP application knows where to send the SIP REGISTER request (3). The LEPP receives the REGISTER and reads the user’s AOR from the To header field. Then the LEPP writes the mapping between user’s AOR and node’s IP address to the P2P network by using the P2P daemon. The P2P daemon writes the mapping with Publish request (4) of P2PP. The P2P network returns 200 OK response (5), by using P2PP, to indicate that the mapping was successfully written. After that the LEPP sends 200 OK (6) response, by using SIP, to the VoIP application.

When the user wants to make a VoIP call to other users in the P2P network, he/she uses his/her VoIP application in a typical manner. The only difference is that the TLD of the domain name in the callee’s SIP URI has to be “.p2p”. When the user has initiated the call, the VoIP application does a DNS query (7) with the domain name part of callee’s SIP URI, and the augmented DNS resolver responds (8) with the IP address of the localhost. After that the VoIP application send SIP INVITE (9) to the LEPP and the LEPP reads the callee’s SIP URI from the To header field of the INVITE. Then the LEPP calculates a hash from the callee’s SIP URI and makes a P2P lookup (10) based on that. The P2P network returns the IP address of the callee in a response (11) and the LEPP forwards the SIP INVITE (12) to the callee’s node. Cacelle accepts the call and return 200 OK response (13) which is then forwarded (14) by the LEPP to the VoIP application. After this, the media exchange can start directly between the caller and the callee. It is noteworthy that signaling sequence above was a simplified version from the actual signaling flow (e.g., 180 Ringing SIP response was omitted).

B. Email Module

The email module in the LEPP enables the decentralization of email applications (such as Mozilla Thunderbird). Our email module is somewhat similar to ePOST (see Section II-B). The difference between ePOST and our framework is that in ePOST email messages are stored in the P2P network and in our framework the P2P network is used just for getting the mapping between the email address and the IP address of the recipient. So, in our framework email messages are stored in local nodes and not in the P2P network. In order to use legacy email applications a user has to manually configure the address of either IMAP or POP3 mail server and the address of the SMTP server to his/her email application. Those addresses can be either the address of the localhost or any hostname where the TLD is ”.p2p”. Furthermore, the user has to input his email address, which ends with ”.p2p”, to the LEPP, so that the LEPP can do the registration to the P2P network.

When the LEPP is started, it registers itself to the P2P network so that other users can reach it, see Fig. 3. The LEPP writes the mapping between the user’s email address and the node’s IP address to the P2P network with Publish request (1) and the P2P network returns 200 OK response (2) to indicate that the mapping has been written successfully.

The user can send an email in a typical manner. When the user sends an email, first the legacy email application does a DNS query (3). Then augmented DNS resolver returns the address of the localhost in a DNS response (4). After that the email application sends the outgoing email to the LEPP with SMTP (5-6). Then the LEPP reads the email address of the recipient from the To header field, calculates a hash from it, and performs a P2P lookup (7) based on the calculated hash. When a 200 OK response (8), carrying the IP address of the recipient, is received by the LEPP, it relays (9-10) the email message to the recipient’s node by using SMTP. Now the email module on the recipient’s node has the email message and it can serve it to the legacy email application of the recipient by using either POP3 or IMAP.

C. Web Module

The web module in the LEPP enables the decentralization of web browsers (such as Mozilla Firefox). Our web module is a bit like a hybrid between the DDNS and Web-over-SFR (see Section II-C). The DDNS is similar in a way that our framework performs P2P lookups based on hostnames like the DDNS does, and our framework is similar to the Web-over-SFR in a way that the web module in the LEPP is acting as a web proxy, like in one implementation of Web-over-SFR. Our framework differs from both, the DDNS and Web-over-SFR, in a sense that it does not require the user to configure the web browser nor the operating system in any way (e.g., the user does not have to configure the HTTP proxy to his/her browser).

If the user wants to serve web pages from the local node, he/she has to manually configure the hostname (e.g., carol.example.p2p) for his/her node, and input the content of the web pages to the LEPP. If a user decides to serve web pages, then the LEPP performs a registration (1-2) to the P2P network each time it is started, see Fig. 4.

When the user wants to fetch a web page, he/she uses his/her web browser in a typical manner. The only difference is that the TLD in the URLs the user types to the web browser has to be ”.p2p” (e.g., URL could be ”www.example.p2p”). After the

---

1See, http://www.mozilla.com
user has typed in an URL to the web browser, the browser does a DNS query (3) with the domain name part of the typed URL, and the augmented DNS resolver responds (4) with the IP address of the localhost. Then the web browser sends an HTTP GET request (5) to the LEPP. The LEPP reads the hostname from the Host header field, calculates a hash from it, and performs a P2P lookup (6) based on the hash. When the 200 OK response (7), carrying the IP address of the target node, is received by the LEPP, it proxies the HTTP GET request (8) to the target node. Once the target node has received the request, it replies with a 200 OK response (9) carrying the requested web page. Finally, the LEPP receives the response and proxies it (10) to the web browser.

IV. PROTOTYPE AND EXPERIMENTATION

We have built a Linux-based proof-of-concept prototype from the decentralization framework. The prototype uses the same algorithms, protocols, and applications as we used for describing the framework in Section III. With the prototype, we wanted to validate that our decentralization framework is implementable in practice, and that the design goals are fulfilled. Topics related to scalability and performance are excluded, because they are rather a matter of a data distribution mechanism than the design of the decentralization framework.

The VoIP module was implemented by modifying the Siproxd\textsuperscript{4}. The main functionality of the modified Siproxd is that it can read callees’ AORs from To header fields and make P2P lookups based on them.

The email module was implemented by modifying the SMTP Proxy 0.2\textsuperscript{5} and by using Postfix and Dovecot\textsuperscript{6}. The main functionality of the modified SMTP Proxy is that it can read recipients’ email addresses from To header fields and make P2P lookups based on them. The Postfix and Dovecot are used for receiving emails and for serving them to legacy email applications.

The web module was implemented by modifying the 3proxy and Simple Web Interface Link Library (SWILL)\textsuperscript{7}. The main functionality of the modified 3proxy is that it can read target hostnames from Host header fields and make P2P lookups based on them. Modified SWILL can register web pages to the P2P network and serve them.

All the P2P lookups that modules require are actually done by the P2P daemon. The P2P daemon was implemented by using XML-RPC library\textsuperscript{8} for the interface towards the modules, and by using our own implementations of Chord DHT and P2PP.

The augmented DNS resolver was implemented by intercepting DNS calls. The system calls are intercepted by using the LD_PRELOAD environment variable together with a C program that processes DNS requests and either forwards them to the system’s DNS resolver or returns the address of the localhost.

The experiments with the prototype validated that the design goals are fulfilled and that the framework can be implemented in practice. The VoIP module was tested with X-Lite, email module with Mozilla Thunderbird, and web module with Mozilla Firefox. It is noteworthy that these applications were completely unmodified. The experiments confirmed that the communication within the P2P network works with unencrypted protocol messages. Furthermore, experiments validated that all the modules were able to share a single data distribution mechanism by using an XML-RPC interface.

V. SECURITY CONSIDERATIONS

The decentralization framework can be supplemented with additional security features. Here we focus especially on one possible supplements: the utilization of a centralized CA.

A centralized CA could be used with the decentralization framework in a way that the CA would distribute certificates to each user of the P2P network. The certificate could contain the SIP URI, email address, URL, node id, and the public key of the user. In practice, this could be achieved by using RELOAD \cite{RELOAD} instead of P2PP as the peer protocol. The RELOAD specifies a certificate-based security model which, for example, guarantees the uniqueness of the SIP URIs, and makes Sybil attacks \cite{Sybil} harder to mount.

VI. FUTURE WORK

Robustness of the decentralization framework could be enhanced. Today, many network users have several nodes, such as laptops, desktop computers, and advanced mobile phones. Nature of some applications, for example, the nature of VoIP applications, is such that users have a natural incentive to keep at least one of their nodes online all the time (i.e., they want to stay reachable). The decentralization framework could utilize this phenomenon. One way to increase the robustness, and to utilize this phenomenon, is to synchronize the application data (e.g., email messages and web content) between all the LEPP instances of a single user. A LEPP instance can quite easily find other online LEPP instances belonging to the same user at the time it registers its modules to the P2P network, because all

\textsuperscript{5}See, http://home.tiscali.cz:8080/~cz210552
\textsuperscript{8}See, http://xmlrpc-c.sourceforge.net
the instances have done the registration using the same name. This kind of synchronization would make the decentralization framework more robust, and would, for example, ensure that all the emails are available on all the nodes the user has.

The existing modules of the LEPP could be enhanced. For example, the VoIP module could be extended to implement some telecom services, such as simultaneous ringing [24], in a decentralized fashion. Furthermore, the LEPP could be extended with new modules. Some online games, for example Quake 3, are implemented by using public servers whose main purpose is to provide a rendezvous service, and therefore they could possibly be decentralized with new LEPP modules.

A comparison between the proposed decentralization framework and the existing centralized infrastructure could be done. The comparison could focus on aspects such as scalability, security issues, incentive structures, performance, and resource consumption.

Interworking mechanisms from the centralized infrastructure to the decentralized framework could be a future work item. Now the decentralization framework allows the message exchange within the P2P network and from the P2P network to the centralized infrastructure, but not the other way around. Possibly a specific gateway nodes which would register their domain names, ending with ".p2p", to the DNS could be used.

VII. CONCLUSION

Even though P2P file sharing applications are widely used today by the users of Internet, still many legacy applications utilize centralized infrastructure. In this paper, we proposed a generic framework for decentralizing legacy applications.

The design goals for our decentralization framework were to make it as generic as possible, support unmodified legacy applications, and to allow the sharing of a data distribution mechanism. We have implemented a proof-of-concept prototype from the decentralization framework. The experiments with the prototype validate that at least three legacy applications can be supported without any modifications. The prototype also validates that a single data distribution mechanism can be shared among the applications.

ACKNOWLEDGMENT

The authors would like to thank Markus Meriläinen, Alfonso Garcia, Antonio Artieme, Diego Cosano, Javier Arias, and Luis Arias for their efforts on implementing the P2P daemon, augmented DNS resolver and some parts of the modules to the prototype. In addition, the authors would like to thank Spanish Ministry of Science and Innovation for grant TEC2008-06539.