Protected Chords in the Web: Secure P2P Framework for Decentralized Online Social Networks

Andreas Disterhöft, Kalman Graffi
Technology of Social Networks, University of Düsseldorf, Universitätsstr. 1, 40225 Düsseldorf, Germany
Email: disterhoeft@cs.uni-duesseldorf.de and graffi@cs.uni-duesseldorf.de

***Abstract*—Online social networks have emerged as a main tool to communicate in the Internet. While centralized solutions are prone to censorship, privacy violations and unwanted marketing of the users data, decentralized solutions, e.g. based on p2p technology, promise to overcome these limitations. One major shortcoming is the need to install additional software, which is progressively not accepted by users which are used to web-based applications. In this paper, we present how WebRTC can be used to implement an installation-free, fully decentralized online social network. With WebRTC, standard browsers can communicate directly, which allows to construct PKI secured p2p overlays with replicated and access-controlled, reliable storage. Evaluation shows that our approach is scalable in terms of the number of users and complies with performance requirements stated to today's social networks.**

I. Motivation

Nowadays, the Internet is dominated by centralized solutions, which unfortunately carry the risk of information leakage, data misuse and spying on the users due to the central access of the provider. Provider of current social networking sites or communication tools, such as Facebook or Skype, have an interest in using the user's data for marketing purposes as well as an obligation to hand over all information and communication details to national security organizations. While in rare cases, such a cooperation might lead to the identification of terrorist activities, the surveillance of millions or even billions of people might lead to the attempt to control these people. In totalitarian regimes, surveillance of the masses and the subsequent selection and punishment of people questioning the regime is an essential building block for manifesting the suppressing power of these regimes.

Thus, alternatives to these centralized, possibly surveilled solutions are needed. Overlay, e.g. Peer-to-Peer (p2p), networks emerge in the Internet as a mean to provide new network functionality to interested nodes, such as information-centric routing or direct user-to-user communication, without the need of central elements. However, current p2p overlays and corresponding distributed mechanisms come with the limitation that potential participants have to install a third-party software in order to join the p2p network. This potential barrier can lead to a reduced growth rate of a system's user base.

With the introduction of WebRTC [1], it is possible to establish an encrypted and direct p2p connection between browser instances. By this, we gain the opportunity to create a secured p2p network based on browsers without the need of installing additional software, because popular browsers already implement WebRTC.

Requirements: As a use case and core functionality, our social networking solution supports the following main features fully distributed: *F1*) a device-independent login *F2*) search and establishing connections to registered users *F3*) buddy list management *F4*) communication via text-based-chat and audio/video streaming and *F5*) a device-independent, distributed storage of the contacts and chat history. As further, non-functional requirements we state: *N1*) Security: First steps should contain user authentication, authorization and access control. *N2*) Full decentralization: any protocol should be planned to be equipped with mechanisms to handle churn and an expected heterogeneity of the user devices' resources.*N3*) Device independence: a user should not be bound to any device. *N4*) Feasible demands: the application should be responsive and consume only small amounts of the user's bandwidth and storage space.

The paper is organized as follows. In Section II, we present a brief taxonomy of distributed online social networks and highlight browser-to-browser-based DHTs for online social networks are new. Section III elaborates on our solution in detail. In the evaluation in Section IV, we analyze our solution under various aspects.

II. Related Work

An overview of the challenges in distributed social networks, which were proposed to alleviate the security and censorship risk of centralized online social networking (osn) sites, is given by Buchegger and Datta [2] and Paul et al. [3]. Two large trends emerged in this research area: private server approaches and p2p-based approaches.

Well-known work in the field of private server approaches are Diaspora [4], FOAF [5] and Persona [6].

These approaches assume users set up web servers, connect them and create a distributed social network in this way. By this, central storage points are omitted and the control on the data remains at the users or their friends but the risk of data misuse and censorship is given, as any web server may be compromised or shut down.

In contrast, p2p overlay networks inherently assume failures of participating nodes and provide a reliable distributed data storage plane with DHTs. Thus, in p2p-based approaches, users are expected to be only temporarily active in the network, permanently online servers are not assumed. The authors in [7], Safebook [8] and GoDisco [9] concentrate on overlay networks mapping the friendship relation, which offers the benefit of efficiency as communication remains mainly local in the overlay as typically friends are interested in the updates and data of their own friends. This first sub-class faces some structural limitations in the case of bootstrapping nodes with few friends or nodes not trusting their friends. Their data availability is at risk as shown by Mega et al. [10]. In order to overcome this problem, the second class creates a DHT that is used as a distributed data plane independent of the friendship relations. This approach is applied by LifeSocial [11], [12], [13], PeerSoN [14], SuperNova [15] and OverSoc [16].

Boldt and Fischer [17] proposed a browser-based p2p network based on the Chord protocol using WebSocket SOCKS5 Proxys and creating the possibility to store encrypted data in the DHT using symmetric cryptography. However, we worked independently in parallel and realized several future work suggestions. Particularly, we implemented a browser-based Chord protocol using WebRTC, asymmetric cryptography, supporting device independence and a basis for osn applications.

III. Our Contribution: A WebP2P-based Framework for Social Networks

In this section, we present our software architecture and the most important protocols. They implement a browser-to-browser based, p2p framework for social networks, in specific implementing a secure buddy list, data storage and personal text / audio / video communication. The requirements stated in Section I, namely *F1-F5* and *N1-N4*, are met, whereby *N4* is proved by the evaluation. Figure 1 shows the modular structure of our software. In the scope of this paper we focus on the most important modules, namely the Web-DHT-Overlay, which provides the basic functions used by upper layers to tackle *F3-F5*, and the realistic user simulation.

1) Web-DHT-Overlay Module: This module is the main part of the software, which contains the p2p overlay and the basic application support. The module is subdivided in the tasks for (a) establishing direct browser-to-browser connections, (b) using these connections to implement a DHT and (c) all security related functions.

For the lowest 1-to-1 communication layer, we use *PeerJS* which encapsulates the WebRTC API. Using this library we establish connections to remote peers, whereas a PeerJS server acts as a broker but subsequent communication is done directly, and start data transmission or audio / video (AV) streaming. Security challenges to overcome were due to that WebRTC-API can be reached via Javascript which defines security risks as addressed by Lawton [18]. We eradicated these issues by using the Google Web Toolkit (GWT) and by following guidelines for this framework. GWT is a framework to build web applications, where Java code is translated to Javascript by emulating Java standard APIs and is used to boost the implementing time. As the framework emulates a list of APIs, there exist limitations, which mostly arise because of the browser environment. For example it is essential to program in a single-threaded event-driven manner. We harnessed the main advantage of GWT to use native Javascript code through the JavaScript Native Interface (JSNI), so an integration of native Javascript libraries became possible. For the higher layers, we introduced a new message type called *Message* which is used to send and dispatch application layer messages, like text messages, buddy requests and AV responses. In case of an incoming message or streaming request, the local node has to decide how to process this request / message (*F4*).

We decided to use Chord with its open source implementation *OpenChord* as the base, which is full decentralized and provides stabilization mechanisms to handle churn (*N2*). Despite the fact that Chord is known for not being churn-resistant, we have chosen to use the basic Chord overlay as it is simple and can be easily modified to fit our needs in the future. Thus, due to our modular structure, we are able to expand, modify and even exchange our overlay in every detail. By applying well-known features, mechanisms and strategies from other overlays we may be able to improve the stability.

Figure 2 shows all modifications on plain OpenChord. We added, removed and modified features with the aim to support browser-to-browser interaction, security, robustness and to provide with the application layer a usable basis for social applications on top.

First of all, the *Chord ID* is now calculated from the users credentials (user name, password), which is a deterministic process. This gives us device independence (*N3*). To do so, we generate an asymmetric elliptic curve cryptography (ECC) key pair from the credentials. The generated public key is now used as the Chord ID, so every message can be encrypted by using the destination's public key and signed by the author. The receiving node can verify the signature. To support this, nodes create self-signed certificates, which are stored on deterministic, well-defined places in the Chord ring *i*) hash(publicKey) and *ii*) hash^j(username). Please note the entry in (i) is unique, so we get the exact search for a given public key. The entries in (*ii*) are shared among the users with the same user name. Iterating through the logical user name list will return the desired contact. Thus, the search for a peer is equivalent to the search for its certificate (*F2*). These deterministic processes imply, however, that a password change is currently not possible.

To preserve the overriding of user certificates and further data elements in the DHT, we implement a distributed identity-based access control mechanism. We introduce three types of DHT entries to provide an identity-based access control: *i*) *unsigned*: a simple unsigned entry which can be modified by every node. Basic attributes: ID, author ID and payload *ii*) *signed*: an entry which contains the basic attributes plus a signature so other nodes can verify it. Only the author of this entry is allowed to modify this entry and *iii*) *signed_encrypted*: this entry contains in addition to the basic attributes a signature but its payload is encrypted by the author's public key. Certificates are stored as signed entries so that only the author should be able to modify the entry. Each node checks whether an incoming store operation is permitted by verifying the author and its signature. In combination with the above mentioned asymmetric cryptography on communication layer these are first steps towards security (*N1*). If a participant decides to leave the network, he revokes his certificates by overriding them with dummy certificates. This is necessary as the logical username list (hash^j(username)) must not be interrupted. Otherwise the username search may become corrupted, however, the public key search would still work fine.

In order to realize an authenticated login to the network we customized the join and create processes. We support both public bootstrap nodes announced by web-servers as well as private bootstrap nodes entered by the users. If the local user wants to register and there is no valid certificate stored, the node tries to insert the certificates at the deterministic, well-defined places mentioned above. If the local user wants to join and his certificate is valid, the node will announce itself as bootstrap node to the web-server. After a successful certificate check the user has finally joined the Chord ring (*F1*).

2) GUI Module Addon - User Simulation: Besides the GUI implementation for desktop and mobile platforms (Figures 3, 4) we implemented the realistic behavior of a simulated osn application user. Those models are taken from Leskovec et al. [19], who modeled the user behavior in an instant messaging application in a planetary-scale. Furthermore, the work from Guha et al. [20] extracted some characteristics for relayed audio video streaming. Both papers formed the basis for our user simulation model, which is used by our evaluation in Section IV.

Attribute	*Value*
Nodes	50 nodes on 13 machines (CPUs: i3-2100, i5 560M, C2Q Q9550)
Behaviour	realistic behaviour (see Section III.2)
Webserver	accessible via www.webp2p.de with StartSSL certificate, receives monitoring information, generates and provides friendship graph, runs PeerJS brokering server
Churn	disabled dynamics
Scenario	get user instance and join public Chord (10 minutes)
	stabilize and synchronize (5 + 8 minutes at maximum)
	add contacts (2 requests per minute)
	start IM / AV using Google Chrome fake AV device (2 hours)
Duration	24 hours simulated in 2 hours

TABLE I
SIMULATION PARAMETERS

IV. EVALUATION

In the evaluation we focus mainly on the performance and cost measures and side effects of this platform while simulating realistic user behavior in a non-dynamic scenario. It includes the following aspects: *i*) Can our system cope with the requirements for an instant messaging (IM) / AV streaming platform? *ii*) What are the traffic costs (maintenance, application, all)? *iii*) What response times do occur for operations and message types? *iv*) How much time does the crypto module take in this process? *v*) What is the average hop count of started operations? All relevant simulation parameters are listed in Table I and the outcome of this scenario is summarized by the next sections. Please note that all figures referenced in this section show the arithmetical average over cumulative data within intervals of 20 seconds and its 95 % confidence intervals.

1) Streaming: Figure 5 shows the fairly low byte rate of 4.3 KByte/s at the end of the scenario. This is mainly due to the fake device provided by Google Chrome which generates low quality dummy streams compared to a real 720p web cam which exhibits a four times higher bandwidth consumption depending on the streamed content (static picture vs. many movements). Another reason for such a low traffic is due to the power-law distributed friend graph, where many nodes only have one to two and a few nodes more than ten friends and our AV streaming scheduler. Declined AV stream requests, which were refused because the requested node is already streaming, are rescheduled with an exponential backoff strategy and a new created request to a random friend is created and sent. Thus, the probability to get a stream started with a node is reciprocal to its number of friends.

2) Bandwidth Usage: Sent bytes per second throughout the whole scenario are shown in Figures 6 and 7. The conspicuous rise at minute 23 is caused by certificate searches, its contact add requests and subsequent conversations, which use FindSuccessor, Message and RetrieveEntries messages. Due to periodically called buddy list savings, which store the full buddy list into the DHT which in turn is replicated, the bandwidth consumption of

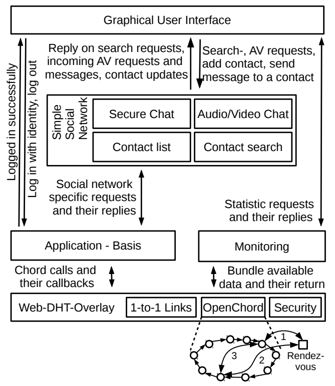
Graphical User Interface
Logged in successfully
Log in with identity, log out
Reply on search requests, incoming AV requests and messages, contact updates
Search-, AV requests, add contact, send message to a contact
Simple Social Network
Secure Chat
Audio/Video Chat
Contact list
Contact search
Social network specific requests and their replies
Statistic requests and their replies
Application - Basis
Monitoring
Chord calls and their callbacks
Bundle available data and their return
Web-DHT-Overlay
1-to-1 Links
OpenChord
Security
1
2
3
Rendez-vous

Fig. 1. Software architecture

Category	*Message type*	*Sent [B/s]*	*[%]*
Streaming	*WebRTC specific*	4300	78.2
Backup Jobs	InsertEntry	207.7	11.3
	InsertReplica	417.2	
Stabilization Jobs	Notify	269.9	7.8
	Ping	157.0	
	FindSuccessor	2.7	
Messages	Message	117.6	2.1
Search / Add	FindSuccessor	22.2	0.5
	RetrieveEntries	3.2	
Other	NotifyCopy	2.7	0.1
	RemoveReplica	0.9	
	Shutdown	0.3	
	Total	5501.4	100

TABLE II
TRAFFIC PER CATEGORY

Message type	*send/receive in avg [s]*	*send/receive in total [s]*
Ping	0.058 / 0.068	79.5 / 94.1
Notify	0.058 / 0.068	79.2 / 93.9
Message	0.058 / 0.068	48.8 / 57.4
FindSuccessor	0.058 / 0.070	10.1 / 12.2
InsertReplica	0.141 / 0.175	4.6 / 5.2
InsertEntry	0.150 / 0.131	2.0 / 2.2
RetrieveEntries	0.117 / 0.100	1.9 / 2.1
RemoveReplica	0.059 / 0.070	0.4 / 0.5
NotifyCopy	0.125 / 0.092	0.4 / 0.4
Shutdown	0.051 / 0.065	0.2 / 0.2

TABLE III
AVG / TOTAL CRYPTO TIME OF CHORD'S COMMUNICATION LAYER (SORTED BY TOTAL)

Layer	*Added*	*Adapted*
Communication	• Browser-to-browser protocol • Audio/video streaming via PeerJS • Application message type • Messages contain Operation ID's	• Node addressing • Removed: Connect messages • Remove: COM protocols
DHT Overlay: Web-Open-Chord	• Certificate infrastructure and management • Application layer messages • Audio/video streaming between peers	• ID out of Identity instead of URL • Join and create process • DHT entries
Security	• Encrypted communication • Operations on certificates • Security in DHT entries	
Monitoring	• Operation pool for Operation ID's	
All	• Event-based callbacks	• Threading removed

Fig. 2. WebChord - Adaptations made to OpenChord

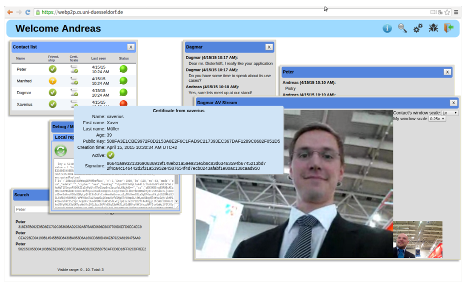

Fig. 3. GUI for desktop PC's.

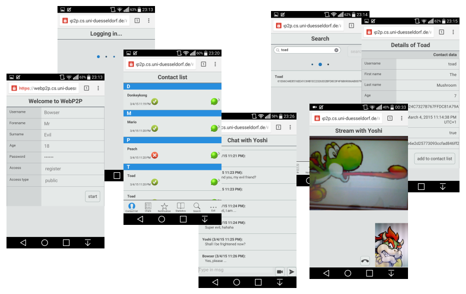

Fig. 4. GUI for mobiles.

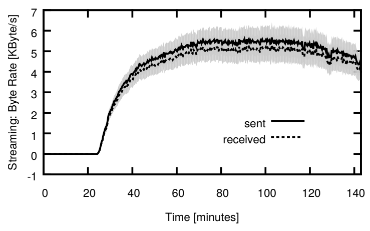

Fig. 5. Traffic of AV streaming

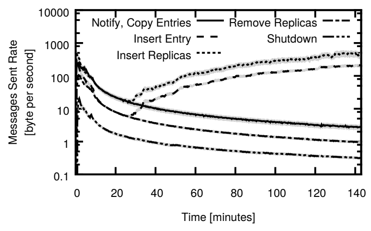

Fig. 6. Traffic of sent messages - Type A

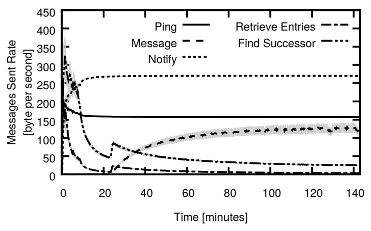

Fig. 7. Traffic of sent messages - Type B

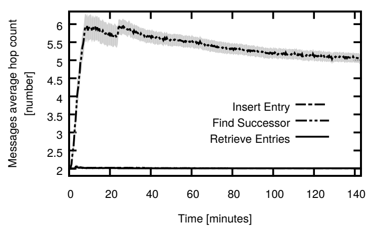

Fig. 8. Hop count per message type

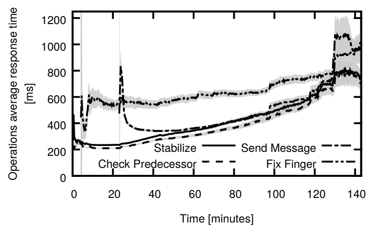

Fig. 9. Average response time - Op. A

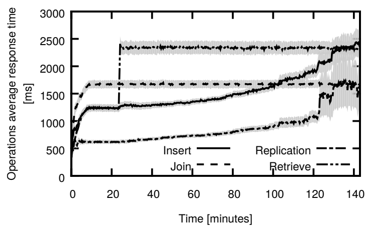

Fig. 10. Average response time - Op. B

InsertEntry and InsertReplica is steadily rising. Hence, at the end, some users have to store large data, which results in a leap and large confidence intervals. This reflects in Table II, where the backup jobs consume 11 % in total, including 7 % solely for the replication. Still, the streaming takes the biggest share with 78 %. Compared to the relatively low share of the application messages (2 %) the stabilization jobs' consumption is the four-fold of it. In our setup a user needs an averaged and aggregated bandwidth of 44 KBit/s for each direction. This is quite low compared to the statistics of OOKLA, where each user in the Internet got a download and upload bandwidth of 23.2 and 10.6 MBit/s, respectively - as of 11.05.2015.

3) Cryptography Time of Communication Layer: Table III presents the time taken by the security layer to process messages sent and received by the communication layer of Chord. This is a crucial metric for the user experience as a high number means a high probability the GUI freezes because those tasks may block the single thread processing the browser's event queue. Summarized, every node needed only 5.8 % of the whole time to process messages in the crypto module. On average, the used CPUs easily dealt with the occurred workload. Weaker CPUs may suffer from split second freezes, thus web workers are advisable in order to tackle this problem.

4) Hop count: Figure 8 demonstrates all routed message types and their average hop count measured by time. As we use recursive routing the FindSuccessor's graph value of 5.02 at the end of the simulation has to be halved in order to get the path length (2.51). Due to the disabled dynamics this complies with and even fall below the theoretical path length of $\frac{1}{2} * \log_2 50 = 2.82$.

5) Response Time: In our scenario for the most message types the hop count is two and the speed of the access is 1 GBit/s, thus the response time is roughly the crypto time plus a small RTT. Figures 9 and 10 show that the response time averaged over all operations is 1.4 seconds. Response times for operations, which a user initiates through a GUI interaction and waits for its response, are most critical, like all those operations, which use the Send Message operation. Those operations only take around one second for the feedback, so users get a reasonably fast feedback on a GUI interaction.

V. Conclusion and Future Work

This paper introduces a way to use a browser-based peer-to-peer network without installing any third-party software. The presented implementation is a modular and secure browser-based social communication platform, which uses the WebRTC standard to realize direct communication via browsers. We use an heavily modified OpenChord version for our p2p overlay, which now offers certificate-based user search, security for communication and DHT entries and a few chat platform features like sending application layer messages and starting audio / video streams. Evaluation shows our platform's resource consumption is low compared to average user's available resources, thus, *N4* is fulfilled.

In the future, we plan to further increase the overall performance of the platform in order to cope with a more dynamic scenario. This contains delta-updates for buddy list savings, the extension to or the use of a more churn-resistant overlay and web workers to process crypto methods on weak CPU's without blocking the GUI. Additionally, we plan to extend the number of users in our scenario by using a distributed testbed like PlanetLab. Also, we consider to perform an evaluation and extension of our security basis with regard to route poisoning and end-to-end encryption.

References

[1] D. Burnett, A. Narayanan et al., "WebRTC 1.0: Real-Time Communication Between Browsers," Tech. Rep., 2013, http://www.w3.org/TR/2013/WD-webrtc-20130910/.

[2] S. Buchegger and A. Datta, "A Case for P2P Infrastructure for Social Networks - Opportunities & Challenges," in *IEEE WONS*, 2009.

[3] T. Paul, B. Greschbach et al., "Exploring Decentralization Dimensions of Social Networking Services: Adversaries and Availability," in *ACM HotSocial*, 2012.

[4] A. Bielenberg, L. Helm et al., "The Growth of Diaspora - A Decentralized Online Social Network in the Wild," in *IEEE INFOCOM*, March 2012.

[5] J. Golbeck and M. Rothstein, "Linking Social Networks on the Web with FOAF: A Semantic Web Case Study," in *AAAI*, 2008.

[6] R. Baden, A. Bender et al., "Persona: An Online Social Network with User-defined Privacy," in *ACM SIGCOMM*, 2009.

[7] G. Mega, A. Montresor et al., "Efficient Dissemination in Decentralized Social Networks," in *IEEE P2P*, 2011.

[8] L. A. Cutillo, R. Molva et al., "Safebook: A Distributed Privacy Preserving Online Social Network," in *IEEE WoWMoM*, 2011.

[9] A. Datta and R. Sharma, "GoDisco: Selective Gossip Based Dissemination of Information in Social Community Based Overlays," in *IEEE ICDCN*, 2011.

[10] G. Mega, A. Montresor et al., "On Churn and Communication Delays in Social Overlays," in *IEEE P2P*, 2012.

[11] K. Graffi, S. Podrajanski et al., "A Distributed Platform for Multimedia Communities," in *IEEE ISM*, 2008.

[12] K. Graffi, C. Gross et al., "LifeSocial.KOM: A P2P-Based Platform for Secure Online Social Networks," in *IEEE P2P*, 2010.

[13] ——, "LifeSocial.KOM: A Secure and P2P-based Solution for Online Social Networks," in *IEEE CCNC*, 2011.

[14] S. Buchegger, D. Schiöberg et al., "PeerSoN: P2P Social Networking: Early Experiences and Insights," in *ACM SNS*, 2009.

[15] R. Sharma and A. Datta, "SuperNova: Super-peers based Architecture for Decentralized Online Social Networks," in *IEEE COMSNETS*, 2012.

[16] D. I. Wolinsky, P. St. Juste et al., "OverSoc: Social Profile Based Overlays," in *IEEE WETICE*, 2010.

[17] D. Boldt and S. Fischer, "Return the data to the owner: A browser-based peer-to-peer network," in *IARIA ICIW*, 2014.

[18] G. Lawton, "Web 2.0 Creates Security Challenges," *IEEE Journal on Computer*, vol. 40, no. 10, 2007.

[19] J. Leskovec and E. Horvitz, "Planetary-scale views on a large instant-messaging network," in *ACM International WWW*, 2008, pp. 915–924.

[20] S. Guha, N. Daswani et al., "An Experimental Study of the Skype Peer-to-Peer VoIP System," in *IPTPS*, 2006.