Scamplon

Brice Nédelec, Julian Tanke, Davide Frey, Pascal Molli, and Achour Mostefaoui

***Abstract*—Recently, WebRTC allowed the connection establishment between web browsers, even with complex network settings such as firewall, proxies or Net Address Translation (NAT). It eases the end-user access to distributed applications. However, several applications require to broadcast messages to a potentially large number of participants. Traditionally, gossiping algorithms handle such broadcast. Unfortunately, current gossiping algorithms raise reliablity and efficiency issues in the WebRTC context, where the connection establishement uses a three-way handshake protocol. Indeed, since the peers cannot afford to have global knowledge of the network membership, they only have a partial view of the latter. However, current approaches either lake of adaptiveness (e.g. CYCLON) or robustness (e.g. SCAMP). In this paper, we introduce SCAMPLON, a random peer sampling protocol (i) which provides a logarithmic partial view size in average to each peer, (ii) which quickly converges to a random graph, (iii) which can handle three-way handshake. Experimentation shows that it outperforms the state-of-the-art approaches both in the traditionnal and the three-way handshake connection set-up.**

I. INTRODUCTION

Recently, WebRTC [1] allowed the connection establishment between web browsers, even with complex network settings such as firewall, proxies or Net Address Translation (NAT). Some attractive applications such as Firefox Hello [2] or WebTorrent [3] demonstrate how video-conferencing or file-sharing directly within browsers becomes simple of access to end-users without third-party providers. However, several distributed applications such as distributed collaborative editing (REF) require the ability to broadcast messages to a potentially large number of participants. Traditionally, gossiping algorithms (REF) achieve such broadcast. Unfortunately, the current gossiping algorithms in a WebRTC network raises reliability and efficiency issues.

Gossiping requires that each participant maintains connections with a uniform sample of other participants [4]. The sample, called partial view, can grow logarithmically compared to the number of participants and yet providing connectivity with high probability. Still, the neighbouring must be constantly renewed in order to prevent network partitions. In WebRTC, establishing a browser-to-browser connection requires a three-way handshake which is much more costly and likely to fail compared to traditional setting. Consequently, existing gossiping algorithms will either be resource intensive, or fail to maintain a connected network.

The CYCLON protocol [5] provides each peer with a fixed size partial view. Each element in this view has an age which allows quick withdrawing of left peers. Nevertheless, the user must foresee the maximum size of the network to set the appropriate view size. As such, the views are generally oversized generating too much resource consumption in a WebRTC network. On the other hand, the SCAMP protocol [6], [7] incrementally builds the partial views of peers at join. Thus, each peer has a partial view size logarithmically growing compared to the global size of the network. Nevertheless, a peer may connect to peers at several hops away from him. Hence, the connections are more likely to fail.

In this paper, we introduce SCAMPLON, a random peer sampling protocol (i) which provides each peer with a partial view logarithmically growing compared to the global network size, (ii) which quickly converges to a random graph, (iii) which can handle three-way handshake. Experiments shows that it outperforms the state-of-the-art approaches both in one-way connections and three-way handshake connections.

The rest of this paper is organised as follow: Section II introduces the necessary background to understand SCAMPLON and highlights our motivations. Section III details the SCAMPLON protocol. Section IV shows the properties of SCAMPLON and compare them to state-of-the-art random peer sampling approaches. Section V reviews the related work. We conclude and discuss about perspective in Section VI.

II. BACKGROUND AND MOTIVATIONS

This section introduces the general outlines of protocols in charge of providing the peer sampling. Then, it details two state-of-the-art approaches that will be useful later on. Finally, we highlight the motivation of our approach.

A. Random peer sampling

Peer sampling protocols [4] provide each peer with a partial view $\mathcal{P}$ of the network membership $\mathcal{N}$. In this paper, we focus on random peer sampling protocols where these latter populate the partial views with peers chosen at random among $\mathcal{N}$ following a uniform distribution. They aim to converge to a random graph, hence, providing robustness and efficient epidemic spreading of messages. A wide variety of gossip-based protocols use random peer sampling (e.g. topology management [8], [9]). Nonetheless, providing each peer with a uniform random sample relying solely on local knowledge remains challenging (REF?). Our SCAMPLON protocol is inspired by two state-of-the-art approaches, namely SCAMP and CYCLON.

CYCLON [5] is a periodic random peer sampling protocol which updates its partial view every interval of time. The partial view has a fixed-size set at start. In this view, each neighbour is associated with an age incremented at each cycle. To update its partial view, CYCLON exchanges a subset of its partial view with one of its neighbour chosen using the age. Assuming a well chosen $|\mathcal{P}|$, CYCLON quickly converges to a random graph (cf. (REF experiments)) and quickly withdraws

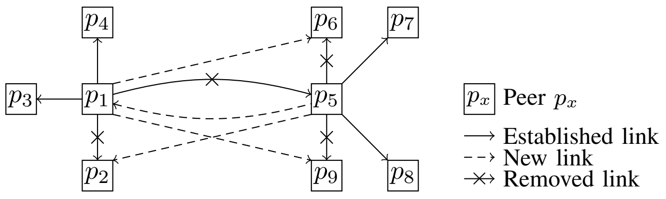

Fig. 1. Example of CYCLON's periodic exchange of partial views. For simplicity sake, we set $|\mathcal{P}|$ to 4 and the exchanges concern only 2 neighbours. Furthermore, while the network contains 9 members, we only show the partial views of p_1 and p_5. Peer p_1 initiates the view exchange with its oldest neighbour p_5, giving $\{p_1, p_2\}$. On receipt, Peer p_5 establishes the connection to p_1 and p_2 and gives away its connection to $\{p_6, p_9\}$. On receipt of this latter, p_1 establishes the connection to Peer p_6 and Peer p_9. Altogether, we cut 4 links to create 4 other links only by using neighbour-to-neighbour interactions.

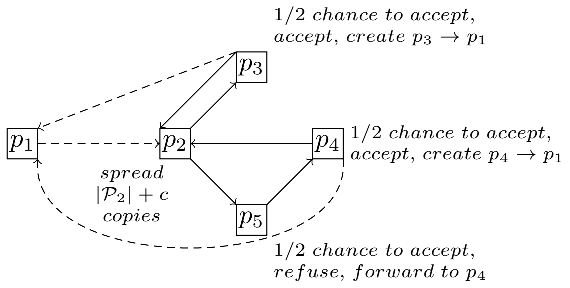
$1/2$ chance to accept, accept, create $p_3 \rightarrow p_1$
p_3
p_1
p_2
p_4
$1/2$ chance to accept, accept, create $p_4 \rightarrow p_1$
spread $|\mathcal{P}_2| + c$ copies
p_5
$1/2$ chance to accept, refuse, forward to p_4

Fig. 2. SCAMP's joining protocol on a small network with an identical legend to Figure 1. Peer p_1 wants to join p_2's network composed of p_2, p_3, p_4, and p_5. Let $\mathcal{P}_x$ denotes the partial view of Peer p_x. Peer p_1 establishes a connection to p_2. Then p_2 copies $|\mathcal{P}_2|$-times the new subscription and sends one to each neighbour. To handle possible failures, c additional copies (here 0 for simplicity sake), are sent to random neighbours. Then, each peer p_x has a probability of $\frac{1}{|\mathcal{P}_x|+1}$ to accept the forwarded subscription, hence, to establish a connection to reach p_1. Otherwise, the peer forwards it to another peer in $\mathcal{P}$ at random. Here, p_3 accepts the subscription, p_5 forwards it to $p4$, and the latter accepts it. The resulting graph is connected.

crashed/left peers. Figure 1 depicts the periodic exchanges of CYCLON. In particular, we can see that peers only communicate in their direct neighbourhood to renew the connections.

SCAMP [6], [7] stands for SCalable Membership Protocol. This protocol links each membership event with an appropriate reaction. The most important event is the joining event which creates a logarithmically increasing number of connections compared to the total network size. Figure 2 depicts the joining protocol of SCAMP. As we can see, the resulting graph is connected. However, the last peer p_1 only has one neighbour in its partial view, meaning that if p_2 crashes, p_1 cannot send messages to the network any more. To avoid that, a periodic re-subscription is necessary. This re-subscription mechanism will likely add new neighbours in the partial view of the newest peers and reorganise the neighbourhood of oldest peer. Just as CYCLON, SCAMP converges to a random graph providing desirable properties.

B. Three-way handshake

The recent WebRTC technology has allowed establishing peer-to-peer connections directly inside modern web browser.

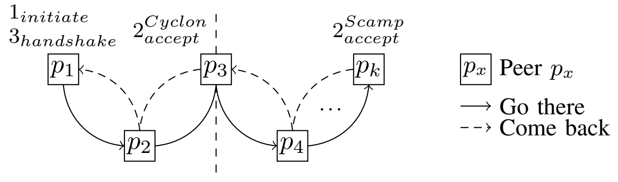

Fig. 3. Handshaking differences of CYCLON and SCAMP. While subscriptions travel from p_1 to p_3 using one intermediate in CYCLON, the subscriptions travel from p_1 to p_k in SCAMP. This difference change the complexity of the two approaches. In particular, the failure probability is much more important in SCAMP. Indeed, links and peers, once employed, must be kept alive until the subscriptions travel back. Otherwise, the handshaking cannot be completed.

In such context, the browsers use a three-way handshake connection establishment instead of a more traditional one-way connection. Thus, a subscription message must be (1) emitted at Peer p, (2) accepted at Peer q, (3) completed at Peer p. Therefore, when a peer needs to establish a connection to another peer, a message must travel back and forth. One could use a dedicated server to establish the necessary dialog. However, this solution does not scale when each peer in the network must repeatedly renew connections over time, which is an expected behaviour of random peer sampling protocols. On the other hand, using the built network itself to properly exchange the subscription messages seems appropriate. Figure 3 depicts how it can be performed in CYCLON and SCAMP. As we can see, CYCLON subscription messages always travel two hops away from their origin, while SCAMP subscription messages can travel longer (k hops where k is in average $log(|\mathcal{N}|)$). This difference deeply impacts performances of SCAMP as detailed hereafter, while CYCLON suffers from other flaws.

C. Motivations

The establishment of connections using three-way handshaking forces a round-trip negotiation much more costly and subject to failure compared to one-way connection. In this context, each connection matters. Hence, we cannot afford oversized partial views like in CYCLON. For instance, considering an application usable by small or large groups of users indifferently, CYCLON will overestimate to adapt to the largest groups. While this is indeed required for these latter, the small groups of users will suffer of increased traffic (oversized partial views maintenance, increased redundancy during the broadcast of messages). If the group is small enough, the graph could be fully connected which is highly undesirable.

As consequence, one would be tempted to use SCAMP since it provides a logarithmically growing number of connections compared to $|\mathcal{N}|$. However, the SCAMP subscription mechanism can create connections several hops away from its origin (cf. Figure 3. Unfortunately, each of these hops is an opportunity of failure.

Let P_f the probability of either the peer or the link between the latter and the next peer crashes/disconnects when it holds the subscription, without possible recovery. Let P_E the probability that a connection establishment cannot be completed. Without three-way handshake, P_E is straightforward:

$$P^{CYCLON}_{E,\,1way} = 1 - (1 - P_f)^2 \quad (1)$$

$$P^{SCAMP}_{E,\,1way} = 1 - (1 - P_f)^{k+1} \quad (2)$$

where k is the number of hops before the acceptance of the subscription, i.e., there are 2 hops at least. Indeed, the first peer cannot directly accept the subscription. On the other hand, in the handshaking context, the subscription must travel back to its origin in order to be completed. As consequence, when a subscription travels through a peer or a link, they are not allowed to fail until the subscription travels back. Thus, we obtain:

$$P^{CYCLON}_{E,\,3way} = 1 - (1 - P_f)^6 \quad (3)$$

$$\begin{aligned} P^{SCAMP}_{E,\,3way} &= 1 - ((1 - P_f)^{2(k+1)}(1 - P_f)^{2k} \ldots (1 - P_f)^2) \\ &= 1 - (1 - P_f)^{k^2+3k+2} \quad (4) \end{aligned}$$

From these analysis, it is worth noting that while CYCLON remains at a same order of magnitude in both cases. On the other hand, SCAMP is dramatically impacted.

To summarise, we would like the best of the aforementioned approaches: a random peer sampling protocol providing a logarithmically growing partial view size compared to the network membership, using neighbour-to-neighbour interactions to establish connections, quickly converging to a random network. Such protocol would allow gossiping on top of WebRTC. Hence, it would open the gate to a wide variety of distributed and decentralised applications requiring a scalable broadcast.

III. SCAMPLON

SCAMPLON[1] is a random peer sampling protocol inspired by both SCAMP and CYCLON. It provides the best of its parents: (i) a logarithmically increasing partial view size compared to the global network size, (ii) and fast convergence to a random graph (iii) using only neighbour-to-neighbour connection establishments. It improves the state-of-the-art approaches in the traditionnal context of one-way connections and greatly outperforms the state-of-the-art [5], [6] in the context three-way handshake connection set up. The latter becomes increasingly important with the appearance of technologies allowing peer-to-peer within modern web browsers. This section details the SCAMPLON protocol through intuitions, examples and algorithms.

SCAMPLON reuses the joining process of SCAMP to incrementally build the network. While such network is not flawless, it has the interesting property of logarithmically increasing the number of connections at each join, compared to the global network size. Still, newest members have a small partial view size, and oldest peers are more clustered. To alleviates these issues, a periodic CYCLON-like protocol takes place in order to balance the partial views, using only neighbour-to-neighbour interactions. This balancing concerns both the partial view sizes, and the uniformity of chosen peers within them. Unfortunately, CYCLON handles fixed-size partial views. Therefore, it must be adapted to handle partial views that grow and shrink as the network dynamically adapts to the membership.

The first issue of CYCLON concerns the partial view sizes. For the recall, the user of CYCLON must foresee the maximum size of the network to set the partial view size, and the size of the subset to exchange from neighbour-to-neighbour. However, in SCAMPLON, a peer with 2 neighbours in its partial view can exchange with a peer with 10 neighbours. Ideally, the resulting size of both partial views would be 6 after the exchange. Yet, reaching this ideal value in one cycle is difficult without the knowledge of each other's partial view size. Instead of acquiring this knowledge, SCAMPLON aims to converge to the ideal value by averaging their view size over exchanges. Therefore, the initiating peer sends $\left\lceil \frac{|\mathcal{P}|}{2} \right\rceil$ members to the chosen peer (chosen by age). The latter sends back $\left\lceil \frac{|\mathcal{P}|}{2} \right\rceil$ members too. After the receipt, both peers remove the sent members and add the received members. It worth noting that the global number of connections after the process must not change. Otherwise, SCAMPLON cannot guarantee that the partial view sizes are logarithmic in average compared to the network size. A remarkable difference regarding CYCLON is that SCAMPLON explicitly allows to have a same neighbour multiple times in a partial view. It impacts on the clustering coefficient. However, the impact is not significant since these artefacts are not numerous because the graph is random, and because they disappear over the SCAMPLON cycles (EXAMPLIFY).

There exists a close relationship between SCAMPLON and the proactive aggregation protocol introduced in [10], [11]. The latter states that, under the assumption of a peer sampling sufficiently random, the mean value μ and the variance σ^2 at a given cycle i are:

$$\mu_i = \frac{1}{|\mathcal{N}|} \sum_{x \in \mathcal{N}} a_{i,\,x} \qquad \sigma^2_i = \frac{1}{|\mathcal{N}|-1} \sum_{x \in \mathcal{N}} (a_{i,\,x} - \mu_i)^2$$

where $a_{i,\,x}$ is the value held by Peer p_x at cycle i. The estimated variance must converge to 0 over cycles. In the SCAMPLON case, the value $a_{i,\,x}$ is the partial view size of Peer p_x at cycle i. Indeed, each exchange from Peer p_1 to Peer p_2 is an aggregation resulting to: $|\mathcal{P}_1| \approx |\mathcal{P}_2| \approx \frac{|\mathcal{P}_1|+|\mathcal{P}_2|}{2}$. Furthermore, each peer performs this exchange protocol $1 + Poisson(1)$ times per cycle (i.e. they initiate one and receive one in average). This relation being established, we are able to conclude that SCAMPLON converges exponentially fast. More precisely, each cycle decreases the variance estimation of the overall system by $2\sqrt{e}$.

ALGORITHM 1 shows the SCAMPLON protocol running at each peer. It is divided between an active thread looping to update the partial view, and the passive thread which provides the reactions to received messages. The functions which are not explicitly defined are the following:

[1]SCAMPLON stands as the contraction of SCAMP and CYCLON.

- $incrementAge(view)$: increments the age of each elements in the view and returns the modified view.
- $getOldest(view)$: retrieves the oldest of peers contained in the view
- $getSample(view, size)$: returns a sample of the view containing $size$ elements.
- $replace(view, old, new)$: replaces in the view all occurrences of the old element by the new element and returns the modified view.
- $rand()$: generates a random number between 0 and 1.

A peer o joining the network will reach a contact peer. The latter will call Function $onContact$ which spreads $(|\mathcal{P}| + c)$ copies of o inside the network. Then, each time a peer receives one of these message, assuming that this peer does not have o already and this peer is not o, it has a probability $\frac{1}{|\mathcal{P}|+1}$ to integrate o in $\mathcal{P}$. Otherwise it forwards the copy to a random neighbour. At this point, o has 1 neighbour in its partial view, and appears $(c + 1)log(|\mathcal{N}|)$ times in partial views of other members. The active thread aims to balance the partial views. Each time Function $loop$ is called, the age of each element in $\mathcal{P}$ is incremented. Then, the oldest peer q is chosen to exchange a subset of their partial view. If Peer q cannot be reached (i.e. it crashed/left), it is removed from the partial view and the operation is repeated. Once the initiating peer p found a reachable peer q, the former selects a sample of its partial view, excluding q and including itself. The size of this sample is half of its partial view, with a minimum of one peer: the initiating peer. The answer of q is of the exact same kind. Since peers can appear multiple times in $\mathcal{P}$, the exchanging peers may send references to the other peer, e.g., Peer o's sample can contain references to q. Such sample, without further processing, would create self-loop (q's partial view contains references to q). To alleviate this undesirable behaviour, all occurrences of the other peer are replaced with the emitting peer. Afterwards, both of them remove the sent sample from their view, remove the chosen neighbour, and add the received sample.

Note that extending the algorithm to provide handshaking is not difficult: it only requires to keep track of the neighbour from where the membership messages arrived, and forward the answer to this neighbour accordingly.

There are few optimisations concerning the establishments of connections. For instance, when a peer p starts an exchange with q, and q has p in its partial view, instead of inverting the link between p and q, and q and p, SCAMPLON does not change them. Another optimisation concerns a peer having a neighbour multiple times in its partial view. While SCAMPLON keeps such information in its partial view, only one connection per neighbour is necessary.

IV. EXPERIMENTATION

In this section, we aim to validate Scamplon by highlighting its properties and compare them to the state-of-the-art random peer sampling protocols. In particular, we inspect 3 properties characteristic of random graphs, namely the *average shortest path length*, the *average clustering coefficient*, and *the partial view size distribution*. Additionnaly, we investigate on *robusteness to random failures*.

Algorithm 1 The SCAMPLON protocol.

```
INITIALLY:
    $\mathcal{P} \leftarrow []$;                     ▷ the partial view, sorted by age
    $p$ ;                                            ▷ identity of the local peer
    $c \leftarrow 2$;                                ▷ additionnal connections when joining

ACTIVE THREAD:
    function LOOP( )                                 ▷ Every $\Delta$ time
        $\mathcal{P} \leftarrow incrementAge(\mathcal{P})$;
        let $q, age \leftarrow getOldest(\mathcal{P})$;
        let $sample \leftarrow getSample(\mathcal{P} \backslash \{q\}, \left\lceil \frac{|\mathcal{P}|-1}{2} \right\rceil) \uplus \{\langle p, 0 \rangle\}$;
        $sample \leftarrow replace(sample, q, p)$;
        $sendTo(q, 'exchange', sample)$;
        let $sample' \leftarrow receiveFrom(q)$;
        $\mathcal{P} \leftarrow ((\mathcal{P} \setminus sample) \setminus \{\langle q, age \rangle\}) \uplus sample'$;
    end function

PASSIVE THREAD:
    function ONEXCHANGE($o, sample$)                 ▷ $o : origin$
        let $sample' \leftarrow getSample(\mathcal{P}, \left\lceil \frac{|\mathcal{P}|}{2} \right\rceil)$;
        $sample' \leftarrow replace(sample', o, p)$;
        $sendTo(o, sample')$;
        $\mathcal{P} \leftarrow (\mathcal{P} \setminus sample') \uplus sample$;
    end function

    function ONSUBS($o$)                             ▷ $o : origin$
        for each $q \in \mathcal{P}$ do $sendTo(q, 'fwdSubs', o)$;
        for $i \leftarrow 0$ to $c$ do
            $sendTo(\mathcal{P}[\lfloor rand() * |\mathcal{P}| \rfloor], 'fwdSubs', o)$;
    end function

    function ONFWDSUBS($o$)                          ▷ $o : origin$
        if $((rand() < (1/(|\mathcal{P}| + 1))) \wedge (o \neq p))$ then
            $\mathcal{P} \leftarrow \mathcal{P} \uplus \{\langle o, 0 \rangle\}$;
        else
            $sendTo(\mathcal{P}[\lfloor rand() * |\mathcal{P}| \rfloor], 'fwdSubs', o)$;
        end if
    end function
```

Experiments are simulations ran on the well-known PeerSim simulator (REF). We implemented Scamp, Cyclon, and Scamplon. The code is available on the Github platform[2]. Most of the experiments involve 100, 1000, and 10000 peers.

A. Clustering coefficient

OBJECTIVE:

DESCRIPTION: The average clustering coefficient measures the connectivity of each peer's neighbourhood in the network.

$$\overline{C} = \frac{1}{|\mathcal{N}|} \sum_{x \in \mathcal{N}} C_x \qquad (5)$$

where C_x is the local clustering coefficient of Peer p_x. The higher the coefficient, the more likely the network contains cliques. These experiments involve 100, 1000, and 10000

[2]http://github.com/anonymous4now

peers. Cyclon is set to be optimal at 1000 peers with $|\mathcal{P}|$ set to 7. The exchanges concern 3 neighbours.

RESULTS:

REASONS:

B. *Average path length*

OBJECTIVE:

DESCRIPTION: The average path length is the average of the shortest path length between peers in the graph. It counts the minimum number of hops to reach a peer from another given peer. The lower the value, the faster a message can reach the whole network.

RESULTS:

REASONS:

C. *Partial view size distribution*

D. *Resilience to failure*

E. *Churn*

F. *Synthesis*

V. RELATED WORK

Currently, the ecosystem of distributed browser-embedded applications is blooming [2], [3], [12]. Indeed, the arrival of peer-to-peer within modern browsers opened the gate to a large variety of real-time distributed web applications, previously quartered to standalone applications. As consequence, they become accessible to a larger audience. A majority of these applications targets a small number of simultaneous users (e.g. Video conferencing [2]) but more recently, applications aiming larger scale have been arising. For instance, WebTorrent [3] is a file-sharing system based on the BitTorrent protocol. Nevertheless, users of this application share static data. As such, the built network could be completely clustered but still working as long as one member in each cluster has all the data. Till now, the class of applications requiring scalable broadcast has been neglected. The goal of SCAMPLON is to fill this gap by providing a scalable membership protocol well-adapted to such context. Using SCAMPLON, the network is connected, yet using few connections. Therefore, the broadcast messages reach all members very efficiently.

To foresee the size of the partial view, one could use a network size estimator. There exists a plethora of network size estimator [10], [13], [14], [15], either using (i) sampling techniques [13], [14], [16] which analyse a network subset and deduce the network size using probabilistic functions, (ii) sketching techniques [15] which use hashing to compress the high amount of data and deduce the network size using the collisions, (iii) averaging techniques [10] which use aggregations that converge over exchanges to a value which depends of the network size. The drawbacks of these approaches lie in their cost (e.g. communication overhead) and/or their assumptions (e.g. a static network). On the other hand, SCAMPLON provides partial views logarithmically growing compared to the network size. Thus, it does not imply any overhead since it converges to a random graph by design. Conversely, SCAMPLON cannot provide a precise estimation of the network size.

VI. CONCLUSION AND PERSPECTIVES

In this paper, we introduced a random peer sampling approach called SCAMPLON. This new approach, inspired of two state-of-the-art approaches, namely SCAMP and CYCLON, provide the best of both worlds: (i) logarithmically growing partial views compared to the global network size, (ii) a quick convergence to a random graph (iii) using neighbour-to-neighbour interactions. As such, SCAMPLON outperforms its parents in the context of 1-way connection establishments. In addition, it greatly outperforms state-of-the-art in the 3-way handshake connection establishment set-up which becomes increasingly important with recent technologies such as WebRTC that allows peer-to-peer connection within modern browser. Since, random peer sampling constitutes the ground of many decentralised applications.

Future work includes an analysis of the chosen size of exchanges in SCAMPLON. Indeed, currently, each peer choose a linear number of neighbours to exchange compared to the size of partial view: $\left\lceil \frac{|\mathcal{P}|}{2} \right\rceil$. Nevertheless, the paper introducing CYCLON shows that a small subset of the partial view is enough to provide a nearly optimal convergence speed. Thus, it lowers the network overhead induced by the partial view exchanges at a relatively small price.

Future work also includes a Javascript implementation using WebRTC. While experimentation of this paper shows the results of Peersim simulations, the real implementation would open the gate to emulations, and even real peer-to-peer decentralised applications (MOAR).

ACKNOWLEDGMENT

REFERENCES

[1] Webrtc. [Online]. Available: http://www.webrtc.org

[2] Firefox hello. [Online]. Available: https://www.mozilla.org/fr/firefox/hello

[3] Webtorrent. [Online]. Available: https://github.com/feross/webtorrent

[4] M. Jelasity, R. Guerraoui, A.-M. Kermarrec, and M. van Steen, "The peer sampling service: Experimental evaluation of unstructured gossip-based implementations," in *Middleware 2004*, ser. Lecture Notes in Computer Science, H.-A. Jacobsen, Ed., vol. 3231. Springer-Verlag, 2004, pp. 79–98.

[5] S. Voulgaris, D. Gavidia, and M. van Steen, "Cyclon: Inexpensive membership management for unstructured p2p overlays," *Journal of Network and Systems Management*, vol. 13, no. 2, pp. 197–217, 2005. [Online]. Available: http://dx.doi.org/10.1007/s10922-005-4441-x

[6] A. Ganesh, A.-M. Kermarrec, and L. Massouli, "Scamp: Peer-to-peer lightweight membership service for large-scale group communication," in *Networked Group Communication*, ser. Lecture Notes in Computer Science, J. Crowcroft and M. Hofmann, Eds. Springer Berlin Heidelberg, 2001, vol. 2233, pp. 44–55. [Online]. Available: http://dx.doi.org/10.1007/3-540-45546-9_4

[7] A. Ganesh, A.-M. Kermarrec, and L. Massoulie, "Peer-to-peer membership management for gossip-based protocols," *Computers, IEEE Transactions on*, vol. 52, no. 2, pp. 139–149, Feb 2003.

[8] S. Voulgaris and M. van Steen, "Epidemic-style management of semantic overlays for content-based searching," in *Euro-Par 2005 Parallel Processing*, ser. Lecture Notes in Computer Science, J. Cunha and P. Medeiros, Eds. Springer Berlin Heidelberg, 2005, vol. 3648, pp. 1143–1152. [Online]. Available: http://dx.doi.org/10.1007/11549468_125

[9] M. Jelasity, A. Montresor, and O. Babaoglu, “T-man: Gossip-based fast overlay topology construction,” *Computer Networks*, vol. 53, no. 13, pp. 2321 – 2339, 2009, gossiping in Distributed Systems. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128609001224

[10] M. Jelasity and A. Montresor, “Epidemic-style proactive aggregation in large overlay networks,” in *Distributed Computing Systems, 2004. Proceedings. 24th International Conference on*, 2004, pp. 102–109.

[11] A. Montresor, M. Jelasity, and O. Babaoglu, “Robust aggregation protocols for large-scale overlay networks,” in *Dependable Systems and Networks, 2004 International Conference on*, June 2004, pp. 19–28.

[12] Sharefest. [Online]. Available: https://www.sharefest.me

[13] A. Ganesh, A.-M. Kermarrec, E. Le Merrer, and L. Massouli, “Peer counting and sampling in overlay networks based on random walks,” *Distributed Computing*, vol. 20, no. 4, pp. 267–278, 2007. [Online]. Available: http://dx.doi.org/10.1007/s00446-007-0027-z

[14] D. Kostoulas, D. Psaltoulis, I. Gupta, K. P. Birman, and A. J. Demers, “Active and passive techniques for group size estimation in large-scale and dynamic distributed systems,” *Journal of Systems and Software*, vol. 80, no. 10, pp. 1639 – 1658, 2007, methodology of Security Engineering for Industrial Security Management Systems. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S016412120700012X

[15] C. Baquero, P. Almeida, R. Menezes, and P. Jesus, “Extrema propagation: Fast distributed estimation of sums and network sizes,” *Parallel and Distributed Systems, IEEE Transactions on*, vol. 23, no. 4, pp. 668–675, April 2012.

[16] E. Mane, E. Mopuru, K. Mehra, E. Mane, E. Mopuru, K. Mehra, and J. Srivastava, “Network size estimation in a peer-to-peer network,” 2005.