Spray: an Adaptive Random Peer Sampling Protocol

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

SPRAY: an Adaptive Random Peer Sampling Protocol

Brice Nédelec
Université de Nantes, LINA
brice.nedelec@univ-nantes.fr

Julian Tanke
Université de Nantes, LINA
julian.tanke@fu-berlin.de

Davide Frey
INRIA Bretagne-Atlantique
davide.frey@inria.fr

Pascal Molli
Université de Nantes, LINA
pascal.molli@univ-nantes.fr

Achour Mostefaoui
Université de Nantes, LINA
achour.mostefaoui@univ-nantes.fr

ABSTRACT

The introduction of WebRTC has opened a new playground for large-scale distributed applications consisting of large numbers of directly-communicating web browsers. In this context, gossip-based peer-sampling protocols appear as a particularly promising tool thanks to their inherent ability to build overlay networks that can cope with network dynamics. However, the dynamic nature of browser-to-browser communication combined with the connection establishment procedures that characterize WebRTC make current peer-sampling solutions inefficient or simply unreliable. In this paper, we address the limitations of current peer-sampling approaches by introducing SPRAY, a novel peer-sampling protocol designed to avoid the constraints introduced by WebRTC. Unlike most recent peer-sampling approaches, SPRAY has the ability to adapt its operation to networks that can grow or shrink very rapidly. Moreover, by using only neighbor-to-neighbor interactions, it limits the impact of the three-way connection establishment process that characterizes WebRTC. Our experiments demonstrate the ability of SPRAY to adapt to dynamic networks and highlight its efficiency improvements with respect to existing protocols.

Categories and Subject Descriptors

C.2.1 [**Computer Systems Organization**]: Computer-Communication Networks—*Network Architecture and Design*

General Terms

Network, algorithm, simulation

Keywords

Large scale distributed applications, random peer sampling, browser-to-browser communication, WebRTC

1. INTRODUCTION

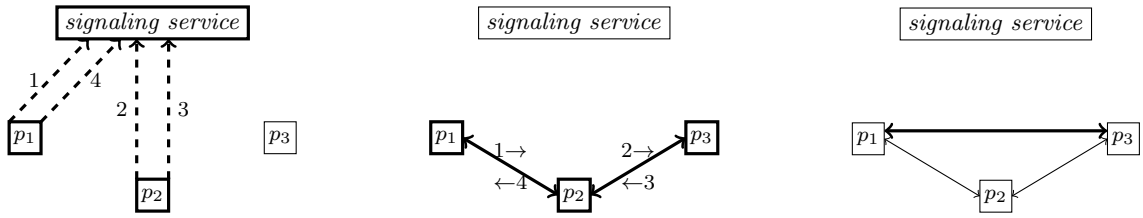
Peer sampling constitutes a fundamental mechanism for many large-scale distributed applications both on the cloud [4] and in a peer-to-peer setting. Services such as dissemination [6, 19], aggregation [12] and network management [13, 21] have been based on peer sampling and the recent introduction of WebRTC[1] opens the opportunity to deploy such applications on browsers that can run on laptops, desktops and mobile devices. In this context, WebRTC drastically simplifies deployment even within complex network systems that utilize firewalls, proxies and Net Address Translation (NAT).

Unfortunately, WebRTC has several constraints that make existing peer sampling services inefficient or unreliable. Browsers can run on small devices in mobile networks. Hence, keeping the number of connections as low as possible is a major requirement. However, peer sampling services such as CYCLON [20] do not adapt the number of connections to the real number of participants. For instance, a user must maintain 10 connections with other remote browsers when only 6 are enough. On the other hand, the peer sampling service SCAMP [10] is adaptive but uses random dissemination paths to establish connections which is much more costly and likely to fail in the WebRTC context.

In this paper, we introduce SPRAY, a random peer sampling protocol inspired by both SCAMP [10] and CYCLON [20]. Compared to the state of art, (i) SPRAY dynamically adapts the neighborhood of each peer. Thus, the number of connections grows logarithmically compared to the size of the network. (ii) SPRAY only uses neighbor-to-neighbor interactions to establish connections. Thus, the connections are established in constant time. (iii) SPRAY quickly converges to a topology exposing properties similar to those of a random graph. Thus, the network becomes robust to massive failures, efficiently disseminates information etc. (iv) In the experimental setup, we show the adaptiveness of SPRAY and highlight its efficiency improvement compared to CYCLON and SCAMP, at the cost of little overhead.

The rest of this paper is organized as follows: Section 2 reviews the related work. Section 3 details the SPRAY protocol. Section 4 shows the properties of SPRAY and compares them to state-of-the-art random peer sampling approaches. We conclude and discuss about the perspective in Section 5.

[1] `http://www.webrtc.org/`



(a) p_1 connects to p_2 using the signaling service. 1: p_1 pushes its offer ticket; 2: p_2 pulls the ticket; 3: p_2 pushes its response; 4: p_1: pulls the response and establishes a bidirectional connection with p_2. p_3 does the same with p_2. Figure 1b depicts the resulting network.

(b) p_1 connects to p_3 using p_2 as mediator. 1: p_1 sends its offer ticket to p_2; 2: p_2 forwards it to p_3 and registers p_1 as the emitter; 3: p_3 sends its response to p_2; 4: p_2 forwards it to the emitter p_1 which connects to p_3.

(c) The resulting network overlay: a fully connected network composed of 3 members.

Figure 1: Creating an overlay network on top of WebRTC.

2. RELATED WORK

WebRTC allows real-time peer-to-peer communication between browsers even when complex network settings such as firewalls, proxies or Net Address Translation (NAT) are involved. However, WebRTC does not manage addressing nor routing. To establish a connection, the browsers exchange offers and acknowledgments through a common mediator, e.g., mails, dedicated signaling services [2], existing WebRTC connections[3] etc. In Figure 1a, p_1 wants to connect to p_2. Therefore, p_1 pushes an offer ticket to a shared signaling service[4]. Peer p_2 pulls the offer, stamps it and pushes it back to the signaling service. Finally, p_1 pulls the stamped ticket and establishes a bidirectional connection with p_2. Identically, p_3 establishes a connection to p_2. We refer to the round-trip procedure as *three-way handshake*. At this point, Peer p_1 is able to establish a connection to p_3 without the mediation of the former signaling service. Instead, it uses p_2 as temporary signaling service. As shown in Figure 1b, Peer p_1 pushes an offer ticket to p_2. As p_2 is already connected to p_3, it forwards the offer to p_3 and registers p_1 as the emitter. Peer p_3 stamps the ticket and sends it back to p_2 who then forwards it back to p_1. Upon receipt, p_1 establishes a bidirectional connection with p_3. Notice that if p_2 crashes during the forwarding process, the connection establishment will fail, even if an alternative route exists as WebRTC does not manage routing.

Using signaling services and existing WebRTC connections allows easy deployment of random peer sampling protocols [11] in browsers that can run on mobile phones or tablets connected to mobile networks. In this context, it is crucial to keep the number of connections as low as possible in order to reduce traffic usage and limit resource consumption.

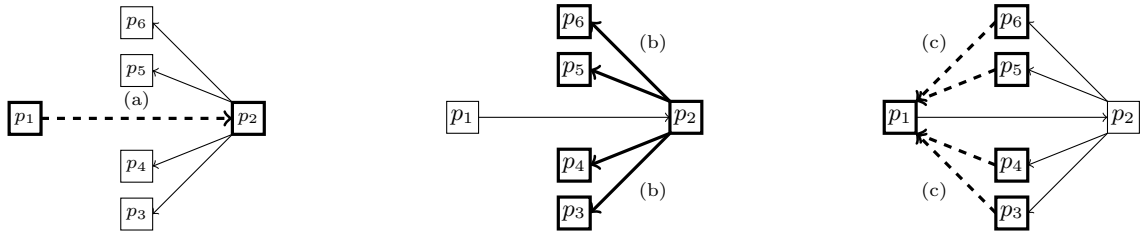
Random peer sampling protocols [11, 14] provide each peer with a partial view $\mathcal{P}$ of the network membership $\mathcal{N}$. They populate the partial views with references to peers chosen at random among $\mathcal{N}$ following a uniform distribution using local knowledge only. Their goal is to converge to an overlay network exposing properties similar to those of random graphs [5]. They efficiently provide connectedness, robustness, information dissemination etc. A wide variety of gossip-based protocols use random peer sampling (e.g. topology management [3, 13, 21]).

The representatives of random peer sampling protocols using a fixed-size partial view [14] are lpbcast [6], Newscast [19], and Cyclon [20]. They have to know *a priori* the maximum network size to set their parameters accordingly. These decisions cannot be safely retracted afterwards. This inflexibility makes it possible to maintains 7 connections in the browser despite requiring only 4, while, in the following moment, it still maintains 7 connections while 10 would be needed. Cyclon's partial views are commonly oversized compared to the actual network size to prevent having too few connections per peer, which, consequently, introduces overhead. When WebRTC is involved, we need a dynamic peer sampling service that is able to adapt to the dynamic number of participants.

Network size estimators can introduce adaptiveness in peer sampling. These approaches either use (i) sampling techniques [8, 15, 16] which analyze a network subset and deduce the network size using probabilistic functions, (ii) sketching techniques [1, 7] which use hashing to compress the high amount of data and deduce the network size using the collisions, (iii) averaging techniques [2, 12] which use aggregations that converge over exchanges to a value which depends on the network size. Unfortunately, while they can be very precise in their estimation, they imply a communication overhead and may have strong assumptions (e.g. random graph topology). However, adaptiveness should introduce a minimum overhead to peer sampling in WebRTC applications.

The sole representative of adaptive-by-design random peer sampling is Scamp [9, 10]. Its interesting property lies in its logarithmically growing partial view sizes meeting the sharp threshold of connectedness of random graphs [5]. Nevertheless, Scamp suffers from other drawbacks. In particular, it systematically disseminates the connections at random. Thus, the originating peer can be several hops away from the arrival peer. In WebRTC, each random dissemination path must be traveled back to finalize the connection es-

[2] `http://peerjs.com/`
[3] `http://ozan.io/p/`
[4] Many signaling services can exist



(a) p_1 contacts p_2 to join the network. p_1 adds p_2 to its neighborhood. p_1 sends its request to p_2.

(b) The $onSubs(p_1)$ event is raised at p_1 which forwards the subscription to p_1's neighborhood.

(c) The $onFwdSubs(p_1)$ event is raised at p_{3-6}. The peers add p_1 to their neighborhood.

Figure 2: Example of the SPRAY's joining protocol.

tablishment, as illustrated in Figure 1. This drastically impacts the SCAMP failure probability of establishing a connection. Let P_f be the probability that an element of the dissemination path (either a peer or a connection) crashes or leaves during a hop of the three-way handshake, without any possible recovery. Let P_E be the probability that a connection establishment cannot be completed. Without three-way handshake, P_E is straightforward:

$$P_{E,\,1way}^{Scamp} = 1 - (1 - P_f)^{k+1} \tag{1}$$

This corresponds to the probability that each element (arc and peer) in the path of size $k+1$ stays alive during their part of the dissemination (i.e., otherwise, they are allowed to crash or leave). In the context of WebRTC, the offer ticket must travel back to its emitter. As a consequence, the elements of the random dissemination path are not allowed to fail until the stamped ticket travels back. We obtain:

$$\begin{aligned} P_{E,\,3way}^{Scamp} &= 1 - ((1 - P_f)^{2(k+1)}(1 - P_f)^{2k} \dots (1 - P_f)^2) \\ &= 1 - (1 - P_f)^{k^2+3k+2} \end{aligned} \tag{2}$$

In other terms, the first chosen arc and peer in the path must stay alive $2k+2$ hops, the second chosen arc and peer must stay alive $2k$ hops etc. The complexity class of the SCAMP failure rate increases leading to a quicker degeneration of the connection count. This behavior endangers the network connectedness, as depicted in Section 4.7.

Building an adaptive-by-design random peer sampling that meets WebRTC constraints raises the following scientific problem:

PROBLEM STATEMENT 1. *Let t be an arbitrary time frame, let $\mathcal{N}^t$ be the network membership at that given time t and let $\mathcal{P}_x^t$ be the partial view of peer $p_x \in \mathcal{N}^t$. A cost-efficient random peer sampling should provide the following best-case properties:*

1. *Partial view size:* $\forall p_x \in \mathcal{N}^t,\ |\mathcal{P}_x^t| = \Theta(\ln|\mathcal{N}^t|)$
2. *Connection establishment:* $O(1)$

The first condition states that the partial view size is relative to the size of the network at any time. It also states that partial views grow and shrink logarithmically compared to the size of the network. The second condition states that each connection establishment requires a constant number of intermediary peers. Since this number is constant, connection establishments do not depend on the network size. Lpbcast, Newscast and CYCLON fail to meet the first condition of the problem statement since they do not adapt their views to the network size. SCAMP fails to meet the second condition of the problem statement since each connection implies an unsafe random dissemination protocol.

3. SPRAY

SPRAY is an adaptive-by-design random peer sampling protocol inspired by SCAMP [10] and CYCLON [20]. It comprises three parts representing the lifecycle of a peer in the network. First, the joining process injects a logarithmically growing number of arcs into the network. Hence, the number of arcs scales with the network size. Next, each peer runs a periodic process in order to balance the partial views both in terms of partial view size and uniformity of the referenced peers within them. Quickly, the overlay network converges to a topology exposing properties close to those of random graphs. Finally, A peer is able to leave at any time without giving notice (equivalent to a crash) while the network properties do not degrade.

The key of adaptiveness consists in keeping a consistent global number of arcs during the whole protocol. Indeed, unlike CYCLON, SPRAY is always on the edge of the optimal number of arcs compared to the network size. Since SPRAY never creates additional arcs after the joining, any removal is a definitive loss. Thus, firstly, SPRAY's joining adds some arcs to the network. Secondly, SPRAY's shuffling preserves all arcs in the network. Thirdly, SPRAY's leaving cautiously removes some arcs, ideally the number of arcs introduced by the last joining peer.

Occasionally, keeping the global number of arcs constant forces the shuffling and the leaving processes to create *duplicates* in partial views. Thus, a partial view may contain multiple occurrences of a particular neighbor. In this paper, we show that the number of duplicates remains low and do not impact the network connectedness.

3.1 Joining

SPRAY's joining algorithm is the only part of the protocol where the global number of arcs in the network can increase. To meet the first requirement of the problem statement, this number of arcs must grow logarithmically compared to the

network size. As in SCAMP, we assume that peers meet this constraint. Therefore, each of them uses this knowledge to propagate the identity of the joining peer. Algorithm 1 describes SPRAY's joining protocol. Line 7 shows that the contacted peer only multicasts the new identity to its neighborhood. Afterwards, to limit the risk of connection failures, each neighbor immediately adds the joining peer to their own neighborhood. This fulfills the second condition of the problem statement. In total, the number of arcs in the network increases of $1+\ln(|\mathcal{N}|)$ using only neighbor-to-neighbor interactions.

Algorithm 1 The joining protocol of SPRAY.

```
1: INITIALLY:
2:     P ← ∅;                                  ▷ the partial view is a multiset
3:     p ;                                     ▷ identity of the local peer
4:
5: EVENTS:
6:     function onSubs(o)                                        ▷ o : origin
7:         for each ⟨q, _⟩ ∈ P do sendTo(q, 'fwdSubs', o);
8:     end function

9:     function onFwdSubs(o)                                     ▷ o : origin
10:        P ← P ⊎ {⟨o, 0⟩};
11:    end function
12:
```

The partial view is a multiset of pairs $\langle n, age\rangle$ which associate to the neighbor n with the age age. The multiset allows managing duplicates and age plays the same role as in CYCLON i.e. it accelerates the removal of crashed/departed peers by shuffling with the oldest neighbors first. The *onSubs* event is called each time a peer joins the network. *onSubs* forwards the identity of the joining peer to all neighbors, independent of the age. The *onFwdSubs* event is called when a peer receives such forwarded subscription. It adds the peer as one of its neighbor with an age set to 0 meaning that it is a brand new reference.

Figure 2 depicts a joining scenario. Peer p_1 contacts p_2 to join the network composed of $\{p_2, p_3, p_4, p_5, p_6\}$. For simplicity sake, the figure shows only the new arcs and the neighborhood of p_1 and p_2. Peer p_1 directly adds p_2 in its partial view. Peer p_2 forwards the identity of p_1 to its neighborhood. Each of these neighbors adds p_1 in their partial view. In total, SPRAY establishes 5 connections and the network is connected.

Unfortunately, the partial views of the newest peers are clearly unbalanced and violates the first condition of our problem statement. The periodic protocol described in the next section will re-balance the partial views.

3.2 Shuffling

Unlike CYCLON, SPRAY shuffles partial views of different sizes. The shuffling aims to balance the partial view sizes and to randomly mix the neighborhood between peers. Nevertheless, the global number of arcs in the network is invariant.

In SPRAY's shuffling protocol, the involved peers send half of their partial view to each other. After integration, they both tend to the average of their partial views and the sum of their partial views stays unchanged. In order to keep the arc number invariant, the partial views of SPRAY are multisets. If a peer receives an already known reference, it still stores it, yet as a duplicate. Thus, the SPRAY's shuffling protocol never increases nor decreases the arcs count.

If duplicates have negative impact on the network properties, most of them disappear after shuffling and they proportionally become negligible as the network grows (see Section 4.6).

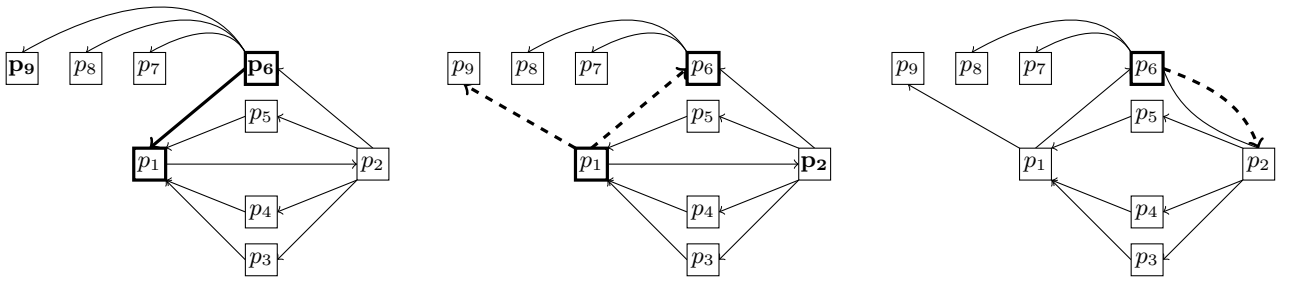
Algorithm 2 The cyclic protocol of SPRAY.

```
1: ACTIVE THREAD:
2:     function loop( )                                          ▷ Every Δ t
3:         P ← incrementAge(P);
4:         let ⟨q, age⟩ ← getOldest(P);
5:         let sample ←
               getSample(P \ {⟨q, age⟩}, ⌈|P|/2⌉ − 1) ⊎ {⟨p, 0⟩};
6:         sample ← replace(sample, q, p);
7:         sendTo(q, 'exchange', sample);
8:         let sample' ← receiveFrom(q);
9:         sample ← replace(sample, p, q);
10:        P ← (P \ sample) ⊎ sample';
11:    end function
12:
13: PASSIVE THREAD:
14:    function onExchange(o, sample)                            ▷ o : origin
15:        let sample' ← getSample(P, ⌈|P|/2⌉);
16:        sample' ← replace(sample', o, p);
17:        sendTo(o, sample');
18:        sample' ← replace(sample', p, o);
19:        P ← (P \ sample') ⊎ sample;
20:    end function
21:
```

Algorithm 2 shows the SPRAY protocol running at each peer. It is divided between an active thread looping to update the partial view, and a passive thread which reacts to an exchange message. The functions which are not explicitly defined are the following:

- *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 floating number between 0 and 1.

In the active thread, Function *loop* is called every Δ time t. First, the function increments the age of each neighbor in $\mathcal{P}$. Then, the oldest peer q is chosen to exchange a subset of its partial view. If Peer q cannot be reached (i.e. it crashed/left), the peer p executes the crash handling function (cf. Section 3.3) and repeats the process until it finds a reachable peer q. Thus, the aging process (which is an inheritance from CYCLON) speeds up the removal of crashed or departed peers. Once it finds a reachable neighbor q, Peer p selects a sample of its partial view, excluding one occurrence of q and including itself. The size of this sample is half of its partial view, with at least one peer: the initiating peer (cf.



(a) Peer p_6 initiates the exchange with p_1 by sending to the latter the multiset $\{p_6, p_9\}$.

(b) Peer p_1 receives the p_6's message. It sends back the multiset $\{p_2\}$ and adds $\{p_6, p_9\}$ to its partial view.

(c) Peer p_6 receives the p_1's response, it adds $\{p_2\}$ to its partial view.

Figure 3: Example of the SPRAY's shuffling protocol.

Line 5). The answer of q contains half of its partial view. 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 behavior, all occurrences of the other peer are replaced with the emitting peer (cf. Line 6, 16). Afterwards, both of them remove the sample they sent from their view and add the received sample.

Figure 3 depicts SPRAY's cyclic procedure. This scenario follows from Figure 2: Peer p_1 just joined the network. Peer p_6 initiates an exchange with p_1 (the oldest among the p_6's partial view). It randomly chooses $\lceil |\mathcal{P}_6| \div 2 \rceil = 1$ peer among its neighborhood. In this case, it picks p_9 from $\{p_9, p_8, p_7\}$. It sends the chosen peer plus its own identity to Peer p_1. In response, the latter picks $\lceil |\mathcal{P}_1| \div 2 \rceil = 1$ peer from its partial view. It sends back its sole neighbor p_2 and directly adds the received neighbor to its partial view. After receipt, Peer p_6 removes the sent neighbors from its partial view, removes an occurrence of p_1, and adds the received peer from p_1. The peers $\{p_6, p_9\}$ compose the p_1's partial view. The peers $\{p_2, p_7, p_8\}$ compose the p_6's partial view.

The example shows that, at first, the initiating peer has 4 peers in its partial view, while the receiving peer has only 1 peer. After the exchange, the former has 3 neighbors including 1 new peer. The receiving peer has 2 neighbors, and both of them are new. Thus, the periodic procedure tends to even out the partial view size of network members. It also scatters neighbors in order to remove the highly clustered groups which may appear because of the joining protocol.

Concerning convergence time of the shuffling algorithm, there exists a close relationship between SPRAY and the proactive aggregation protocol introduced in [12, 18]. It 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:

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

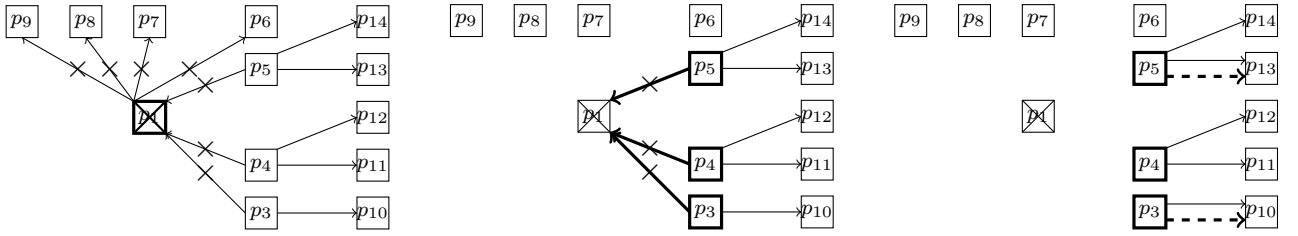
where $a_{i,x}$ is the value held by Peer p_x at cycle i. The estimated variance must converge to 0 over cycles. In other terms, the values tends to be the same over cycles. In the SPRAY case, the value $a_{i,x}$ is the partial view size of Peer p_x at cycle i. Indeed, each exchange from Peer p_1 to Peer p_2 is an aggregation resulting to: $|\mathcal{P}_1| \approx |\mathcal{P}_2| \approx (|\mathcal{P}_1| + |\mathcal{P}_2|) \div 2$. Furthermore, at each cycle, each peer is involved in the exchange protocol at least once (they initiate one), and in the best case 1+Poisson(1) (they initiate one and, in average, each peer receives another one). This relation being established, we know that SPRAY's partial view sizes converge exponentially fast to the global average size. Additionally, we know that each cycle decreases their variance in overall system at a rate comprised between $1 \div 2$ and $1 \div (2\sqrt{\mathrm{e}})$.

The shuffling algorithm provides adaptiveness at the cost of duplicates. Averaging the partial view sizes over exchanges provides a quick convergence to a network topology where the partial views are balanced.

3.3 Leaving and crashing

In SPRAY, peers can leave the network without notice. We make no distinction between node departures and crashes, but the protocol must react to both of them. Without such a reaction, the network could collapse due to an over zealous removal of arcs. When a peer joins the network, it injects $1 + \ln(|\mathcal{N}|)$ arcs. Nevertheless, after few exchanges, the partial view of the joining peer becomes populated with more neighbors. Then, if this peer leaves, it removes $\ln(|\mathcal{N}|)$ arcs from its partial view, and another $\ln(|\mathcal{N}|)$ arcs from peers which have this peer in their partial views. Therefore, without any crash handler, we remove $2\ln(|\mathcal{N}|)$ connections instead of $1 + \ln(|\mathcal{N}|)$. To alleviate this issue, each peer that detects a crash may reestablish a connection with anyone in its neighborhood (which will spread in the network over the exchanges). The probability of reestablishing a connection is $1 - 1 \div |\mathcal{P}|$. Since $|\mathcal{P}| \approx \ln(|\mathcal{N}|)$ peers have the crashed peer in their partial view, it is likely that all of them will reestablish a connection, except one. Therefore, when a peer leaves, it approximately removes the number of connections it injected when it joined.

Algorithm 3 shows the manner in which SPRAY deals with departures and crashes. Function *onPeerDown* shows the reaction of SPRAY when the peer q is detected as crashed or departed. A first loop counts the occurrences of this neigh-



(a) Peer p_1 crashes.

(b) The peers p_{3-5} notice that they cannot reach p_1 anymore.

(c) The peers p_3 and p_5 choose to establish a duplicate with one of their existing neighbor.

Figure 4: Example of SPRAY's crash/leaving handler.

Algorithm 3 The crash/departure handler of SPRAY.

```
1: function onPeerDown(q)                    ▷ q: crashed/departed peer
2:     let occ ← 0;
3:     for each ⟨n, age⟩ ∈ P do                          ▷ remove and count
4:         if (n = q) then
5:             P ← P \ {⟨n, age⟩};
6:             occ ← occ + 1;
7:         end if
8:     end for
9:     for i ← 0 to occ do                    ▷ probabilistically duplicates
10:        if (rand() > 1/(|P|+occ)) then
11:            let ⟨n, _⟩ ← P[⌊rand() ∗ |P|⌋];
12:            P ← P ⊎ {⟨n, 0⟩};
13:        end if
14:    end for
15: end function

16: function onArcDown(q, age)  ▷ q: arrival of the arc down
17:    P ← P \ {⟨q, age⟩};
18:    let ⟨n, _⟩ ← P[⌊rand() ∗ |P|⌋];
19:    P ← P ⊎ {⟨n, 0⟩};                  ▷ systematically duplicates
20: end function
```

bor in the partial view, and removes all of them. Then, the second loop probabilistically duplicates the reference of a known peer. The probability depends of the partial view size before the removals.

Figure 4 depicts the SPRAY's crash/leaving handler. The scenario follows from prior examples after few other exchanges. Peer p_1 leaves the network without giving notice. With it, 7 connections are down. Peers p_3, p_4, and p_5 have the crashed/left peer in their partial view. Peer p_5 has $1 - 1 \div |\mathcal{P}_5| = 2 \div 3$ chance to replace the dead connections. In this case, it duplicates the connection to p_{13}. Identically, p_3 and p_4 detect the crash/leaving and run the appropriate operation. Only p_3 duplicates one of its connection. In total, 5 connections have been removed.

The example shows that some peers reestablish connections if they detect a dead connection. The probability depends on the partial view size of each of these peer. On average, one of these peers will likely remove the arc while the other peers will duplicate one of their existing arcs. In this case, Peer p_1 injected 5 connections when it joined. It removes $7 - 2 = 5$ connections when it leaves. The global number of connections remains logarithmic with respect to the number peers in the network. Nevertheless, we can see that connectedness is guaranteed with the high probability implied by random graphs. Indeed, if Peer p_1 is the sole bridge between two clusters, adding arcs is not enough to ensure connectedness.

Algorithm 3 also shows that SPRAY distinguishes peer crashes and arc crashes. Indeed, Function *onArcDown* deals with connection establishment failures. In this function, the failing arc is systematically replaced with a duplicate. Therefore, the arc count stays invariant even in presence of connection establishment failures. The distinction between the functions *onPeerDown* and *onArcDown* is necessary because the former is supposed to remove a small arc quantity over departures, contrarily to the latter. Without this small removal, the global arc count would grow unbounded with network turnover.

In the context of WebRTC, SPRAY calls the *onArcDown* function when a connection establishement fails. SPRAY calls the *onPeerDown* function when the connection was established once but the neighbor is not responding anymore.

4. EXPERIMENTATION

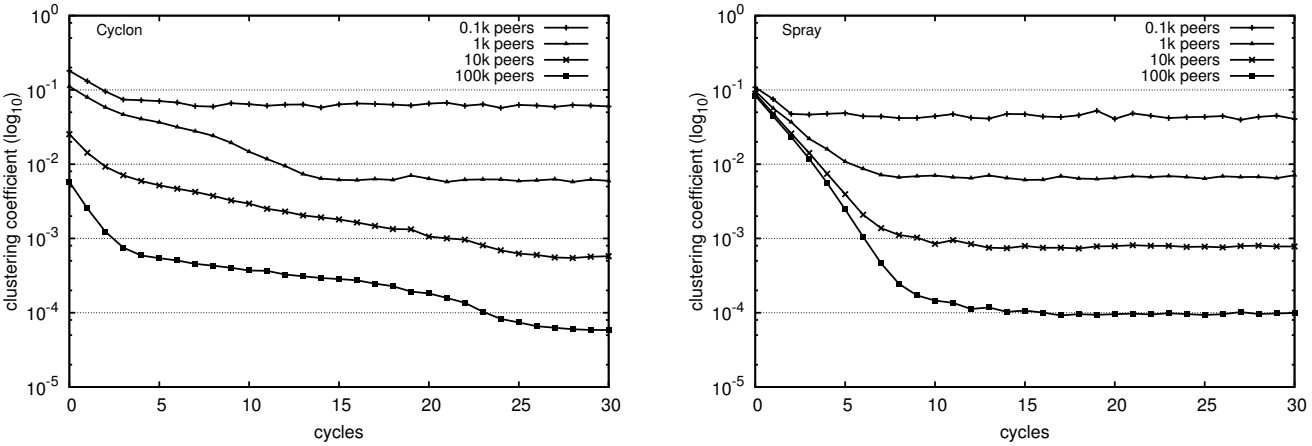
In this section, we evaluate how adaptiveness of SPRAY impacts common metrics of peer sampling performance including clustering coefficient, average shortest path length, indegree distribution, arc count, and connected components. We use CYCLON as a baseline with a fixed-size view for experiments relative to adaptiveness. We expect similar behaviors when the network size is optimal for CYCLON. We expect SPRAY to save resources when CYCLON is oversized. Also, SPRAY should be more robust when CYCLON is undersized. Finally, we expect SPRAY to keep a negligible number of duplicates in its partial views. We use SCAMP as a baseline for experiments relative to connection failures. Unlike SCAMP, we expect SPRAY to tolerate connection failures.

The experiments run on the PEERSIM simulator [17]. The code of the random peer sampling protocols CYCLON, SCAMP, and SPRAY is available on the Github platform[5].

4.1 Clustering coefficient

OBJECTIVE: To observe how adaptiveness impacts on clustering and convergence time.

[5] `https://github.com/justayak/peersim-spray`



(a) Clustering coefficient of CYCLON.

(b) Clustering coefficient of SPRAY.

Figure 5: The x-axis denotes the elapsed time in cycles while the y-axis denotes the $\log_{10}$-scaled clustering coefficient.

DESCRIPTION: The average clustering coefficient $\overline{C}$ measures the connectivity of each peer's neighborhood in the network.

$$\overline{C} = \frac{1}{|\mathcal{N}|} \sum_{x \in \mathcal{N}} C_x \tag{3}$$

where C_x is the local clustering coefficient of Peer p_x. The runs concern 0.1k, 1k, 10k and 100k peers. The representative of fixed-size approach is CYCLON which is optimally configured for 1k peers: its partial views are set to $\ln(1000) \approx 7$ neighbors. During gossip exchanges, the peers using CYCLON shuffle 3 out of their 7 neighbors. Thus, CYCLON is oversized for 0.1k peers and undersized for 10k peers and 100k peers.

RESULTS: Figure 5 shows that CYCLON starts with a lower clustering coefficients than SPRAY. Still, SPRAY converges faster than CYCLON. Furthermore, when the number of peers in the network grows, the convergence time of CYCLON suffers heavily. On the contrary, SPRAY converges very quickly independently of the network size. Figure 5 also shows that both approaches converge to a low clustering coefficient which is characteristic of random graphs. Nevertheless, CYCLON and SPRAY do not reach the same values after convergence. Except when CYCLON is optimally configured, SPRAY's values are either below (when CYCLON is oversized) or above (when CYCLON is undersized). Overall, this shows that SPRAY is 1. faster to converge to a stable clustering coefficient 2. reflecting the needs of the network membership. This impacts both load-balancing and robustness to churn (when peers join and leave the network freely).

REASONS: CYCLON starts with a lower clustering coefficient because each peer performs random walks to advertise themselves in the network. Hence, the starting overlay is already slightly balanced when the simulation starts the shufflings. On the other hand, a newcomer peer in SPRAY only advertises itself to the neighborhood of its contact peer. Therefore, the network overlay starts strongly unbalanced, independently of the network size. Still, CYCLON converges more slowly than SPRAY because of its fixed-size partial view and the size of the shuffle. When they are *a priori* configured, they constitute a constraint to the convergence speed. The clustering coefficient measures how much the neighborhood of each peer is connected to the rest of the network. It directly depends of the partial view size of each peer which, in CYCLON, is fixed. Thus, when the peers number is multiplied by 10, the clustering coefficient after convergence is divided by 10. On the other hand, the peers using SPRAY have variable-size partial views that carefully reflect the network size with a logarithmic growth. Thus, when the network has 1k peers, the partial view size adapts to this network size. This explains the slightly lower clustering coefficient of SPRAY on this run (SPRAY 7.4 vs CYCLON 7). By extending the reasoning, this also explains why SPRAY yields lower values when CYCLON is oversized, and why it yields higher values when CYCLON is undersized.

4.2 Average shortest path length

OBJECTIVE: To observe how adaptiveness impacts on the average shortest path length, i.e., on the efficiency of information dissemination.

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. It basically represents the traveling time of any information to reach all the peers at least once. We average the path length on a small subset of the network membership and run 100 times the simulation on SPRAY to avoid any side effects due to randomness. We also run the simulation on different configurations of CYCLON targeting different optimal network size. CYCLON set with the partial view size of 7 roughly targets 1.1k peers. CYCLON set with the partial view size of 9 roughly targets 8.1k peers. CYCLON set with the partial view size of 11 roughly targets 60k peers. In all these simulations, we perform the measurements after convergence. The checkpoints for the measurements are 0.1k, 0.5k, 1k, 5k, 10k, 50k, and 100k peers.

RESULTS: Figure 6 shows that both CYCLON and SPRAY have an average shortest path length relatively small. Thus, the information can disseminate to all the network very

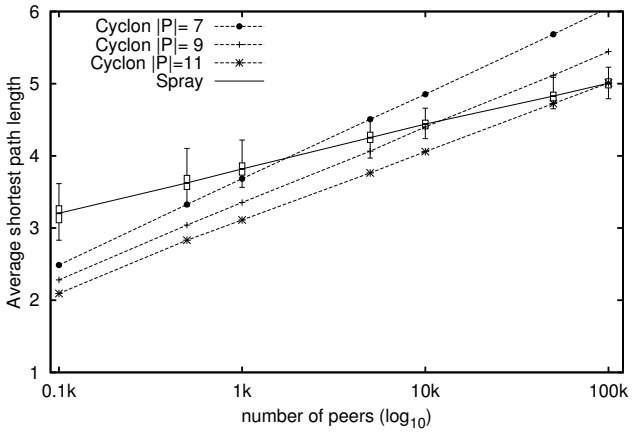


Figure 6: The average shortest path length of SPRAY and CYCLON. The x-axis denotes the number of peers in the network on a $\log_{10}$ scale (from 100 to 100k peers) while the y-axis denotes the average shortest path length of the network.

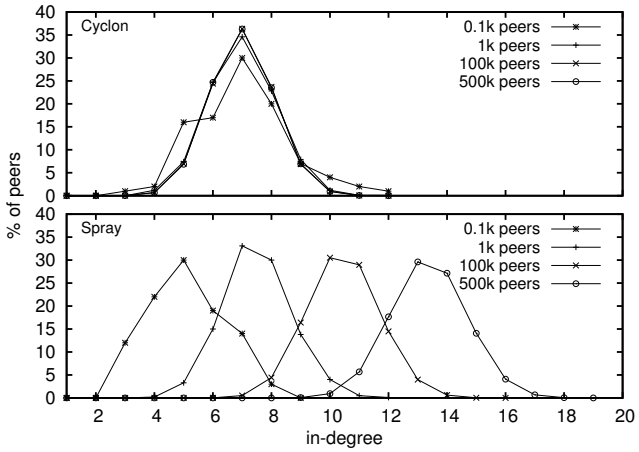


Figure 7: The in-degree distribution of CYCLON and SPRAY. The x-axis denotes the in-degree in number of nodes while the y-axis indicates the percentage of peers with such in-degree. The top figure is dedicated to the runs concerning CYCLON while the bottom figure concerns SPRAY.

quickly. Figure 6 also shows that, each run of CYCLON taken alone can be divided in three parts compared to SPRAY. First, an oversized CYCLON disseminates the information faster than SPRAY. Then, SPRAY and CYCLON are equivalent where the latter is optimally configured. Finally, SPRAY yields better results. Yet, overall, SPRAY scales better than CYCLON since the gradient of the former is lower than any configuration of the latter one.

REASONS: We perform all the measurements after convergence where the network overlay is closely related to random graphs. In such graph, the diameter and average shortest path length stay relatively small, as the resulting values shown in Figure 6. The second observation concerns each CYCLON configuration compared alone with SPRAY. While an oversized CYCLON is much better connected into the graph and thus yields a lower average path length than SPRAY, as soon as it is undersized, SPRAY is, thanks to larger partial views, better connected into the graph. Consequently, it yields the shorter average path length. SPRAY scales better than any configuration of CYCLON because it always follows the optimal value.

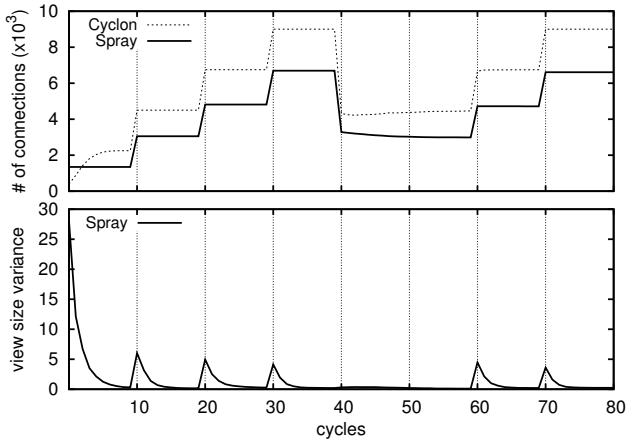
4.3 In-degree distribution

OBJECTIVE: To observe how adaptiveness impacts the in-degree distribution, i.e., the load-balancing among peers.

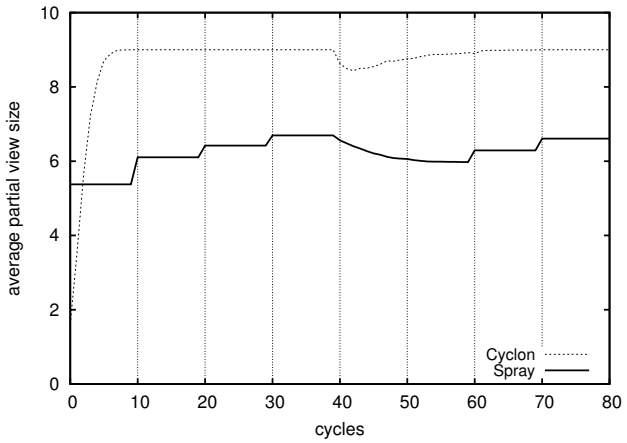
DESCRIPTION: The in-degree of a peer shows how well this peer is represented in others' partial view. The in-degree distribution of the network can highlight the existence of weakly connected peers and strongly connected hubs. It has a direct impact on robustness. In this experiment, the fixed-size partial view approach is CYCLON. It is configured with partial views of size 7 which is optimal for a network of roughly 1100 peers. For all the experiments, we perform the in-degree measurements after convergence. The measurements concern networks with 0.1k, 1k, 100k, 500k peers.

RESULTS: Figure 7 shows the in-degree distribution of CYCLON and SPRAY. In the top figure, we observe that the distribution and the degrees of CYCLON are identical, independently of the network size. Thus, the distribution of 0.1k peers is identical to the 500k one with a mean value of roughly 7 with a strong peak on this value. On the other hand, the bottom figure shows that the distribution of SPRAY follows average partial view size which follows the network size growth. Figure 7 also shows that peers are very gathered around the mean partial view size. For instance, for the run with 500k peers using SPRAY, the mean value for the in-degree is 13.37 and 88 percents of the peers have an in-degree comprised between 12 and 14 included. It means that the load is well balanced among peers. Since peers are equally important in term of connectedness, the network is robust to failures.

REASONS: Once configured, CYCLON must handle any number of peers in the network with a fixed-size partial view. Proportionally, the number of times that a particular peer is referenced does not change compared to the network size. Indeed, the number of arcs that a new peer brings to the network when it joins constitutes that many arcs targeting it after the shuffling rounds. Since the partial view size is constant, the in-degree of peers stays stable. On the other hand, in SPRAY, each joining peer brings a increasing number of arcs in the network. Thus, the in-degree of each peer grows reflecting the network size. Hence, the distribution in the bottom figure shifts slowly to higher in-degree values as the network size grows. SPRAY does not peak on a particular value as CYCLON because the average partial view size for a particular network size falls in-between integer values. For instance, if the average partial view size is 6.5, then half of them will have a size of 6 while the other half will have a size of 7. Such network is robust to failure because no peer is more important than other in term of connectedness. Therefore, if some random peer crashes, it will not affect the network as much as if a strongly connected peer crashed.



(a) The x-axis denotes the elapsed time in cycles. The upper graph y-axis shows the number of total connections in the overlay while the lower graph y-axis shows the variance σ^2 of the partial view sizes in the network.



(b) The x-axis denotes the elapsed time in cycles. The y-axis denotes the average partial view size.

Figure 8: CYCLON (partial view size configured to 9) and SPRAY in a dynamic network. 2.5k peers join the network at cycles 0, 10, 20, and 30. Then 5k peers leave at cycle 40. Finally 2.5k peers join at cycles 60 and 70. The final network contains 10k members.

4.4 Dynamic network

OBJECTIVE: To show the impact of adaptiveness when the network size changes over time.

DESCRIPTION: In this experiment we focus on a dynamic network where peers can join and leave. The runs involve CYCLON and SPRAY. CYCLON's configuration targets roughly 8.1k peers. Thus, it is oversized compared the network size during the simulation (maximum 1k peers). During the first half of the experimentation, 250 peers are added 4 times successively by intervals of 10 cycles each. Thus, the network size goes from 0 to 1k peers in 40 rounds. Then, half of the network leaves without giving notice (500 peers). Finally, 250 peers join two additional times. The final network contains 1k members. The measurements concern 1. the number of connections in the network over cycles, 2. the variance of the partial view sizes over cycles (cf. Section 3.2), 3. the average partial view size of peers.

RESULTS: Figure 8 shows the result of the experiment. The x-axis represents the cycles (i.e. the arbitrary unit time frame). The top part of Figure 8a shows the number of connections established in the network (scale $\times 10^3$) while its bottom part shows the variance in the partial view size of the members. Concerning SPRAY, we can see that at each batch of joining, the connection number grows to reflect the needs of the new network membership. The observation is consistent with the variance measures. Indeed, at each batch of insertions, the variance suddenly grows. Then, it exponentially decreases and converges to zero in less than 10 cycles. The variance is higher when the network size is lower. For instance, the first 250 peers lead to the highest variance. At the 40^{th} cycle, half of the peers leave/crash. Approximately half of the connections are directly removed without disturbing the variance of partial views. The 10 following cycles show a slight decrease of arcs. Then new members are introduced in the network yielding the same results as the earlier joins. CYCLON exposes an identical behavior. Nevertheless, the number of arcs is invariably higher than the SPRAY one (from 1000 to 2500 additional connections). Figure 8b shows the average partial view size of SPRAY and CYCLON. As expected, CYCLON immediately converges to the configured partial view size (9 neighbors). On the other hand, SPRAY's partial views logarithmically grow while the network grows. When the removals occur at cycle 40, the peers using CYCLON remove the dead arcs while refilling their partial view until they reach the configured partial view size. SPRAY only remove the arcs to reflect the departed peers. At the end, the SPRAY partial views contain in average 6.6 neighbors (recall, $\ln(1000) \approx 6.9$).

REASONS: Since the partial views of SPRAY adapt themselves to the network size, the number of connections grows as the network membership grows. The peaks in variance correspond to the joining parts of the experiment. The disparity comes from the fact that new peers arrive in the network with a small partial view. The peaks are smaller when the network is larger. Indeed, the peers - which were already network members before the new arrivals - had a few cycles to exchanges and even out their partial views. As consequence, it lessens the weight of joinings. The removal of 500 peers does not disturb the variance since each crashing/leaving peer is chosen at random. Thus, no peers suffer more of these removals than others. The slightly decreasing number of arcs after the removal is due to peers realizing that some arcs are dead, leading to a probabilistic removal (cf. Algorithm 3).

4.5 Massive failures

OBJECTIVE: To show that both SPRAY and CYCLON are equally robust to massive failures.

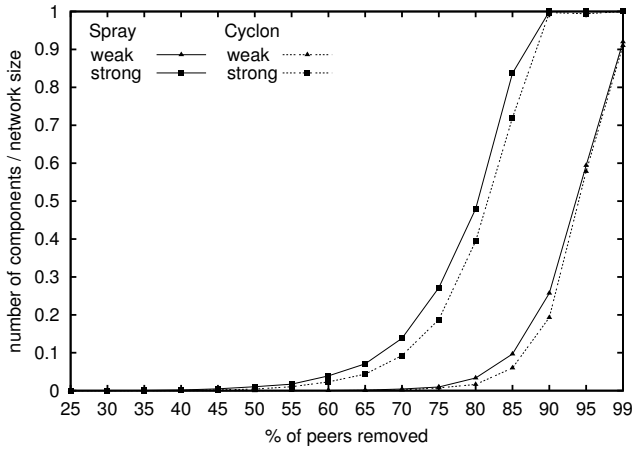


Figure 9: Robustness of CYCLON and SPRAY to massive failures. The x-axis denotes the percentage of peers removed at once in a network containing 10k members. The y-axis denotes the number of components over the current network size (after the removals). The measurements concern the weak and strong components which basically means the number clusters in undirected or directed graph respectively.

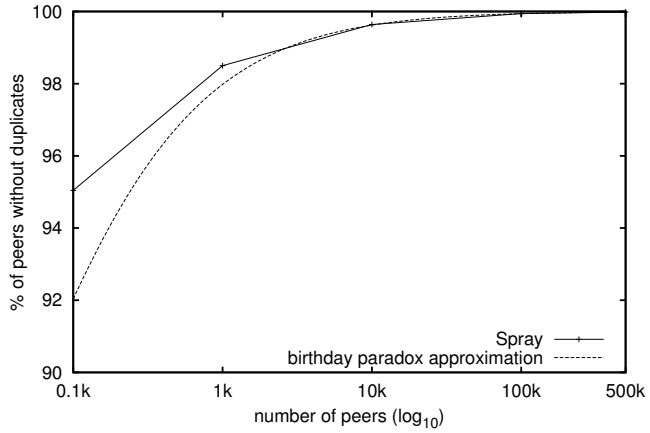


Figure 10: Duplicates in networks of different size: the $\log_{10}$-scaled x-axis denotes the network size while y-axis denotes the proportion of peers without any duplicates in their partial view.

DESCRIPTION: Counting the strong components in a network allows estimating the area that are reachable by the information dissemination protocols. For instance, there are 2 strong components if a part of the network can reach another part and the converse being false. Counting the weak components in a network allows estimating the point where the network is still in a repairable state, i.e., after some shufflings, the network will converge to an overlay where the information dissemination is able to reach all members again. CYCLON has a partial view size set to 9. The network contains 10k members. We perform the removals after the approaches converged to a stable overlay network. We remove the batch of peers at once, from 25 to 95 percents of peers every 5 percents, i.e., 16 runs for each approach. We perform a last measurement at 99 percents. We perform the measurements immediately after the removals. The removals concern a percentage of the peers chosen at random among the 10k members.

RESULTS: Figure 9 shows the ratio of strong/weak components over the network size after removals. First, the figure shows that both the random peer sampling protocols, SPRAY and CYCLON, suffer from deteriorated behavior at high removal percentages, CYCLON being slightly better in this term. Figure 9 shows that the information dissemination (strong components) starts to slowly degrade at 45 percents, and quickly degrade at 70 percents. Fortunately, Figure 9 also shows that the approaches are able to recover from such clustering until high removal rate. Indeed, the weak components start to increase at 70 percents, meaning that some part of the network are completely disjoint and beyond repair.

REASONS: The random peer sampling approaches CYCLON and SPRAY expose very similar results because CYCLON's configuration targets a network size of 10k peers, while SPRAY adjusts itself automatically to this network size. Therefore, the arc number in the approaches are close from being identical CYCLON is slightly better because, in this case, it has more arcs (due to SPRAY's randomness) and because it does not have duplicates (SPRAY contains a small amount of duplicates in the partial view, cf. Section 4.6). It requires a high amount of removals to endanger the information dissemination protocol's ability to reach all peers. Indeed, since all peers are equally important in the network (cf. Figure 7), removing a particular peer does not greatly affect the whole network. The network membership protocol is able to repair the topology later than the information dissemination starts to fail because the direction of arcs is not as important as in the latter. Indeed, CYCLON and SPRAY are still able to repair a network until parts of the network become disjoint, i.e., there is no arc between these parts of the network.

4.6 Duplicates in partial views

OBJECTIVE: To show that a small proportion of peers contain duplicates in their partial view.

DESCRIPTION: Using SCAMP as random peer sampling protocol, we measure the amount of peers which have a partial view containing at least one duplicated reference. We perform the measurements on networks containing 0.1k, 1k, 10k, 100k, and 500k peers. We measure the number of duplicates after convergence. We put this in relation with a theoretical approximation from the birthday paradox. The probability of a peer to not have duplicates is approximately:

$$1 - (1 - \exp(\frac{-\ln(|\mathcal{N}|) * (\ln(|\mathcal{N}|) - 1)}{2 * |\mathcal{N}|})) \qquad (4)$$

RESULTS: Figure 10 shows the proportion of peers using a partial view containing duplicates. As we can observe, there always exist partial views with at least one duplicate. The proportion is more important when the network size is small (e.g. 5 percents for 0.1k peers), and it becomes a minor overhead when the network size is larger (e.g. less than 1 percent for 10k peers). The birthday paradox approximation seems to follow very closely the experimental results. It empirically

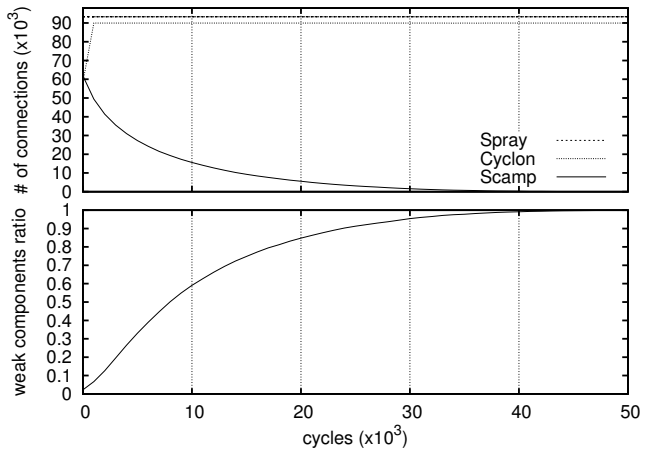


Figure 11: CYCLON, SCAMP, and SPRAY in network subject to failures in the connection establishments. The x-axis denotes the elapsed time in cycles (10^3-scaled). The y-axis of the top figure denotes the global number of arcs (10^3-scaled). The y-axis of the bottom figure denotes the ratio of weak components over the current network size.

confirms that there exist a relation between the duplicates and the birthday paradox. The proportion of peers without duplicates tends to 100 percents as the network size grows.

REASONS: As the network grows, the chances of a particular peer to have at least twice the reference to another peer becomes smaller. Indeed, while the network grows linearly, the number of references to a particular peer only grows logarithmically. Nevertheless, the birthday paradox reminds that this proportion is not as small as it seems to be.

4.7 Failures in connection establishment

OBJECTIVE: To show that SPRAY does not suffer from failures in connection establishments, contrarily to SCAMP.

DESCRIPTION: We measure both the arc count and the number of weak components in the network. The simulations involve CYCLON (configured with partial view containing 9 neighbors targeting a network of roughly 8100 peers), SCAMP[6], and SPRAY. They run over 50k cycles. The network initially contains 10k members. To establish a connection, we use the WebRTC three-way handshake, i.e., the initial peer emits an offer ticket, the arrival peer stamps the ticket, the initial peer finalizes the connection using the stamped ticket (cf. Section 2). The probability that the ticket fails to arrive to its destination is set to 10^{-3} for each hop.

RESULTS: Figure 11 has two parts. The top figure shows the arc count of the random peer samplings while the bottom figure shows the weak components of the network. First, we observe that, as expected, the arc count of CYCLON and SPRAY stays constant over cycles: 90k and 93k arcs for CYCLON and SPRAY respectively. Second, we observe that SCAMP suffers from the failures on connection establishments. It directly impacts on the connectedness of the network represented by the weak components ratio. The network of SCAMP quickly degrades.

REASONS: The arc count of both CYCLON and SPRAY remains constant over time but for different reasons. In CYCLON, the shuffling protocol makes sure that the partial view is filled to its maximum. Thus, when it uses a not successfully established connection to perform an exchange, it simply discards the connection, knowing that the next exchange is likely to overfill its partial view. In SPRAY, when a connection fails to establish and the protocol tries to use it for an exchange, it will replace this arc with another known reference. Thus, the arc count stays constant and the shuffling protocol makes sure that duplicates disappear over time (the arc moves to another peer where it is not a duplicate). In SCAMP, the connections are much more likely to fail than in the aforementioned protocols. Indeed, contrarily to these latter, SCAMP does not cautiously establish connections with the neighbors of its neighbors. Each hop of its ticket dissemination is an opportunity of failure. Since there is no routing in such network, the only way for a stamped ticket to come back to its emitter is the path it traveled in the first place. Hence, all peers belonging to the path must stay alive until the stamped ticket comes back in order to consistently forward it. Furthermore, its periodic protocol starts with an immediate cutting of the incoming arcs of the initiating peer because it assumes that each connection spread in the network will establish. Since it does not, the peer eventually becomes disconnected. Also, when its neighbors execute the periodic protocol, they delete their reference in its partial view. In such case, the peer becomes disconnected and partitions quickly appear.

5. CONCLUSION AND PERSPECTIVES

WebRTC opened a new playground for large-scale distributed applications deployed on a network of browsers. Browsers as infrastructure ease the deployment of large-scale distributed applications for end-users. As a core component of many large-scale distributed applications, we pointed out current peer-sampling protocols' limitations in term of adaptivity or reliability.

In this paper, we described SPRAY, an adaptative-by-design random peer sampling approach designed to fit the WebRTC constraints. SPRAY provides: (i) logarithmically growing partial views reflecting the global network size, (ii) constant time complexity on connection establishments using solely neighbor-to-neighbor interactions,

In experiments, we demonstrated how SPRAY adaptiveness improves random peer sampling performances when network size is changing. In particular, the average shortest path length scales better, the in-degree evolves with the network size, and it converges faster. We also demonstrated that SPRAY stays robust to massive failures. SPRAY and CYCLON are quite similar when the network size is optimal for CYCLON. However, SPRAY saves connections when CYCLON is oversized and is more robust when CYCLON is undersized. Adaptiveness comes at the price of duplicates in the partial views. However, the simulations supported by theoretical analysis shows that the number of duplicates remains very

[6]A modified version of SCAMP whose periodic protocol works properly when there is no connection failures. Available at `https://github.com/justayak/peersim-spray`

low and becomes negligible in large networks.

Future work includes a Javascript implementation of SPRAY. An in-browser implementation opens the gate to emulations, and even real peer-to-peer distributed and decentralized applications.

Future work also includes investigations on topology managers such as T-Man [13] or Vicinity [21]. Indeed, they traditionally rely on random peer sampling approaches using fixed-size partial view. Thus, they maintain a fixed-size view of their most closely related neighbors using a ranking function. With SPRAY, it is possible to extend their behavior to use dynamic partial views. If the view size could adapt to the size of a cluster (e.g. if the topology creates disjoint clusters), it would improve the traffic, robustness, etc.

Acknowledgments

This work was partially funded by the French ANR project SocioPlug (ANR-13-INFR-0003), and by the DeSceNt project granted by the Labex CominLabs excellence laboratory (ANR-10-LABX-07-01).

6. REFERENCES

[1] 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*, 23(4):668–675, April 2012.

[2] F. Blasa, S. Cafiero, G. Fortino, and G. D. Fatta. Symmetric push-sum protocol for decentralised aggregation. In *Proceedings of AP2PS 2011, the Third International Conference on Advances in P2P Systems*, pages 27–32. IARIA, November 2011. ISBN: 9781612081731.

[3] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. *SIGCOMM Comput. Commun. Rev.*, 34(4):15–26, Aug. 2004.

[4] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: Amazon's highly available key-value store. *SIGOPS Oper. Syst. Rev.*, 41(6):205–220, Oct. 2007.

[5] P. Erdős and A. Rényi. On random graphs i. *Publ. Math. Debrecen*, 6:290–297, 1959.

[6] P. T. Eugster, R. Guerraoui, S. B. Handurukande, P. Kouznetsov, and A.-M. Kermarrec. Lightweight probabilistic broadcast. *ACM Trans. Comput. Syst.*, 21(4):341–374, Nov. 2003.

[7] P. Flajolet, É. Fusy, O. Gandouet, and et al. Hyperloglog: The analysis of a near-optimal cardinality estimation algorithm. In *Proceedings of the 2007 International Conference on Analysis of Algorithms*, 2007.

[8] A. Ganesh, A.-M. Kermarrec, E. Le Merrer, and L. Massoulié. Peer counting and sampling in overlay networks based on random walks. *Distributed Computing*, 20(4):267–278, 2007.

[9] A. Ganesh, A.-M. Kermarrec, and L. Massoulié. Scamp: Peer-to-peer lightweight membership service for large-scale group communication. In J. Crowcroft and M. Hofmann, editors, *Networked Group Communication*, volume 2233 of *Lecture Notes in Computer Science*, pages 44–55. Springer Berlin Heidelberg, 2001.

[10] A. Ganesh, A.-M. Kermarrec, and L. Massoulié. Peer-to-peer membership management for gossip-based protocols. *Computers, IEEE Transactions on*, 52(2):139–149, Feb 2003.

[11] M. Jelasity, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. The peer sampling service: Experimental evaluation of unstructured gossip-based implementations. In H.-A. Jacobsen, editor, *Middleware 2004*, volume 3231 of *Lecture Notes in Computer Science*, pages 79–98. Springer-Verlag, 2004.

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

[13] M. Jelasity, A. Montresor, and O. Babaoglu. T-man: Gossip-based fast overlay topology construction. *Computer Networks*, 53(13):2321 – 2339, 2009. Gossiping in Distributed Systems.

[14] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. Van Steen. Gossip-based peer sampling. *ACM Transactions on Computer Systems (TOCS)*, 25(3):8, 2007.

[15] 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*, 80(10):1639 – 1658, 2007. Methodology of Security Engineering for Industrial Security Management Systems.

[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.

[17] A. Montresor and M. Jelasity. Peersim: A scalable P2P simulator. In *Proc. of the 9th Int. Conference on Peer-to-Peer (P2P'09)*, pages 99–100, Seattle, WA, Sept. 2009.

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

[19] N. Tölgyesi and M. Jelasity. Adaptive peer sampling with newscast. In H. Sips, D. Epema, and H.-X. Lin, editors, *Euro-Par 2009 Parallel Processing*, volume 5704 of *Lecture Notes in Computer Science*, pages 523–534. Springer Berlin Heidelberg, 2009.

[20] S. Voulgaris, D. Gavidia, and M. van Steen. Cyclon: Inexpensive membership management for unstructured p2p overlays. *Journal of Network and Systems Management*, 13(2):197–217, 2005.

[21] S. Voulgaris and M. van Steen. Epidemic-style management of semantic overlays for content-based searching. In J. Cunha and P. Medeiros, editors, *Euro-Par 2005 Parallel Processing*, volume 3648 of *Lecture Notes in Computer Science*, pages 1143–1152. Springer Berlin Heidelberg, 2005.