Short: Gossip-Based Sampling in Social Overlays

Mansour Khelghatdoust(✉) and Sarunas Girdzijauskas

Royal Institute of Technology (KTH), Stockholm, Sweden
{mansourk,sarunasg}@kth.se

Abstract. Performance of many P2P systems depends on the ability to construct a random overlay network among the nodes. Current state-of-the-art techniques for constructing random overlays have an implicit requirement that any two nodes in the system should always be able to communicate and establish a link between them. However, this is not the case in some of the environments where distributed systems are required to be deployed, e.g., Decentralized Online Social Networks, Wireless networks, or networks with limited connectivity because of NATs/firewalls, etc. In this paper we propose a gossip based peer sampling service capable of running on top of such restricted networks and producing an on-the-fly random overlay. The service provides every participating node with a set of uniform random nodes from the network, as well as efficient routing paths for reaching those nodes via the restricted network.

Keywords: Peer sampling · Social overlay · Gossip · Random overlay

1 Introduction

At the heart of many gossip protocols, lies a *Peer Sampling Service*, which provides each node with a continuously changing random samples to ensure correctness of the protocols. The assumption of the existing gossip based sampling services, such as [3,6], is that a node can directly contact any other node from its sampled set. However, some applications do not allow such communication freedom between nodes. E.g., nodes behind firewalls, NATs or strictly friend-to-friend online social networks, where communications are limited to immediate friends only, mainly for privacy reasons. In this paper, we provide a solution for executing gossip based sampling service on restricted networks. We ensure that every node is provided not only with the set of random nodes but, crucially, with the routing directions towards these random nodes. I.e., each node maintains routing paths using only the available paths of the underlying restricted network. The main contribution of the paper is a novel on-the fly path pruning technique which exploits the local knowledge of the restricted network at the nodes that forward the gossip messages. The main target of this protocol is particularly for push based gossip applications where nodes update their state only if they have been selected as sample by any participating node to receive information.

G. Noubir and M. Raynal (Eds.): NETYS 2014, LNCS 8593, pp. 335–340, 2014.
DOI: 10.1007/978-3-319-09581-3_26

2 Solution

We model the underlying restricted overlay as an undirected graph with unweighted edges. We assume every node has knowledge (IP address) of its immediate neighbors and neighbors-of-neighbors. We also assume non-malicious behavior of the nodes, i.e., the nodes are willing to act as relay for message forwarding. Apart from knowledge of restricted overlay, nodes keep a fixed-sized and continuously changing cache of C entries of paths towards samples. Similar to [3], in order to acquire truly random samples at each node, nodes periodically exchange with each other the subsets of their caches, called *Swapping Cache*. In particular, in each round every node selects a copy of the longest waiting entry e from the cache, contacts it through routing path and sends to e a copy of swapping cache. Also, it receives a random subset of e's cache entries. Each node then updates its cache with the exchanged set. More precisely, each node updates its cache by replacing entries that are selected as swapping cache more (propagated more) in its cache with the received entries. In our experiments we show that such shuffling strategy converges very fast to random samples at each node and accordingly, a random overlay is continuously being constructed on the fly.

However, since cache entries to the sampled nodes indicate paths which always start at the source node, after swapping, cache paths do not indicate a path from recipient node to the sampled node. A naive way to solve this issue is to merge reversed path from source node to destination node with the path towards the sampled node in the swapped cache. The problem of this approach is that such paths grow prohibitively large and do not scale. We propose an algorithm to prune swapping cache paths upon forwarding gossip messages at each node by exploiting the local knowledge of nodes and discovering shortcuts locally. In this algorithm, source node, destination node and relay nodes (nodes within path) that are involved in a gossiping round execute this algorithm once they have received swapping cache. Every node parses the current path, and prunes it if it can construct a shorter path by using knowledge of its own routing tables or the tables of its neighbors. The message is continued to be relayed through the updated path. The details of the algorithm is given in (Algorithm 1).

Since the underlying restricted overlay can be arbitrary, the resulting routing paths will inevitably exhibit all range of lengths, and can be as short as one hop. Such variation in routing path lengths imply that the communication times between nodes will vary greatly during the exchange process. This in turn will create a bias in selection of random nodes. To this end, we introduce a delay mechanism and define two system parameters called α (maximum threshold of path length) and β (maximum delay). Recipient nodes reject paths with larger length than α to ensure having short length paths. Furthermore, length of gossiping rounds are equalized using β that is preferably equivalent to α. In other words, a gossiping round is postponed to a time that is obtained from the difference between β and the length of the selected path.

From practical perspective, each entry other than path consists of two variables. $WaitingTime$, represents time that entry is waiting to be selected for

Algorithm 1. PATH CONSTRUCTION

```
1: function RECONSTRUCT PATH(path)
2:     resultPath = emptyList
3:     if self.Id is sourceNode then          ▷ Current node is gossip round initiator
4:         resultPath.addFirst(self.Id)
5:         return resultPath
6:     end if
7:     for all id ∈ path.reverse() do
8:         if self.isNeighbor(id) then      ▷ id is immediate neighbor of current node
9:             resultPath.addFirst(id)
10:            break
11:        else if self.isTwoHopNeighbor(id) then          ▷ id is two-hop neighbor
12:            resultPath.addFirst(id)
13:            resultPath.addFirst(self.getNeighbor(id))
14:            break
15:        else
16:            resultPath.addFirst(id)
17:        end if
18:    end for
19:    if self.Id is relayNode then      ▷ Current node is within path acting as relay
20:        resultPath.addFirst(self.Id)
21:    end if
22:    return resultPath
23: end function
```

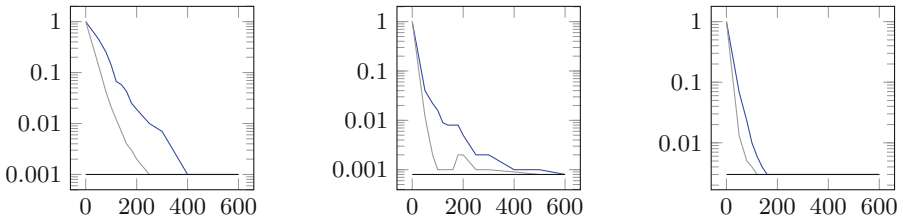
gossiping. *Swapped*, denotes the number of times that the entry was selected as one of the swapping cache entries in current node. The protocol is performed by letting the initiating peer P execute the following 9 steps:

1. Increase by one *waitingTime* of all entries.
2. Select copy of entry Q with the highest *waitingTime* from the cache, and copy of S − 1 other random entries as swapping cache.
3. Increase by one the *swapped* field of all selected S − 1 entries within cache.
4. Set *waitingTime* entry Q within cache to zero.
5. Execute path construction algorithm for all entries of the swapping cache.
6. Wait w.r.t delay, send updated cache to next node of the path towards Q.
7. Receive from one of its social neighbors of reverse path a subset of no more than S of Q's entries and execute path construction algorithm for them.
8. Discard the entries with a path longer than α (Maximum path length).
9. Update P's cache, by *firstly* replacing the same entries already existed with longer path, (if any), and *secondly* replacing entries with the highest *swapped*.

On reception of a swapping cache request, node Q randomly selects a copy of subset of its own entries, of size S, execute step 3 for S entries, executes step 5 and sends it to one of its social neighbors in the constructed reverse path, execute step 7, insert sender entry to the received swapping cache and executes steps 8, 9.

Table 1. DATASET

Data set	$\|V\|$	$\|E\|$	Type	Diameter
Wiki-Vote	7066	103663	Social	7
AstroPh	17903	197031	Collaboration	14
Facebook	63391	817090	Social	16



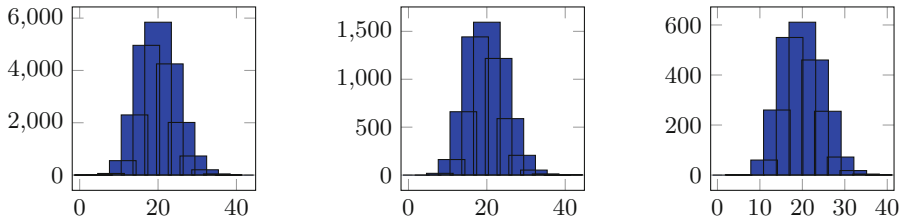
(a) AstroPh (b) Facebook (c) WikiVote

Fig. 1. The figure shows clustering coefficient (CC). X and Y axis-es show cycle and average CC respectively. In blue and gray diagrams both α and β are set 2d and d respectively. Black line represents CC of random graph (Color figure online).

Any relay node involved in gossiping, execute path construction algorithm and cooperate in building reverse path.

3 Experiments

We implemented the algorithm on PEERSIM [5]. The properties of the graphs are given in Table 1. The clustering coefficient (CC) of a node is formulated by division of C over N − 1 that C is cache size and N is network size ($\frac{C}{N-1}$). We calculate it to ensure that CC of resulting overlay is equivalent to random overlay. Neighbors in random overlay are target nodes of paths within caches. Two scenarios are executed (α = d, β = d; α = 2d, β = 2d) where d is diameter of the network. (C = 20, S = 5). As Fig. 1 exhibits, CC of the graphs converge to random graph ensuring global randomness. Larger value for α increases the speed of convergence but gives better local randomness. In another experiment we evaluate In-degree distribution of nodes to see whether the sampling bias is removed in random graph. We calculate degree distribution of target nodes of paths over random graph. The ideal case is to have a degree distribution with low standard deviation. It ensures an unbiased sampling independent of node degrees in social graph. In Fig. 2, the results show a normal distribution in which 70 % of nodes have in degree 20 ± 20 %, a value between 16, 24 (C = 20). In the extended version of our work [4] we perform experiments with churn and show that our algorithm is robust to churn and correlated failures.



(a) Facebook (b) AstroPh (c) Wiki-Vote

Fig. 2. In-degree distribution. X-axis shows degree. Y-axis shows nodes count.

4 Related Work

In [1], a random walk based approach is proposed for sampling on restricted networks. Every node periodically initiates a Maximum Degree random walk, passing its own id and the random walk's mixing time as parameters and using reverse random walk as routing to construct view for keeping samples. This protocol assumes network size and maximum degree are known. So, it needs extra protocols to obtain them. It requires large mixing time and produced average path length is large. It is suitable for small size networks but might not scale in large networks. The idea of constructing a virtual overlay network on top of static networks has already been used in VRR [2]. It is a network routing protocol inspired by overlay routing algorithms in Distributed Hash Tables (DHTs) but it does not rely on an underlying network routing protocol. VRR routes using only fixed location independent identifiers that determine the positions of nodes in a virtual ring. Each node maintains a small number of paths to its virtual neighbors that can be used to forward messages between any pair of nodes. VRR has different usage from our algorithm. It solves routing problem on link layer networks while our algorithm addresses random sampling problem on such networks.

References

1. Bar-Yossef, Z., Friedman, R., Kliot, G.: RaWMS - random walk based lightweight membership service for wireless ad hoc networks. ACM Trans. Comput. Syst. (TOCS) **26**(2), 5 (2008)
2. Caesar, M., Castro, M., Nightingale, E.B., O'Shea, G., Rowstron, A.: Virtual ring routing: network routing inspired by DHTs. ACM SIGCOMM Comput. Commun. Rev. **36**, 351–362 (2006)
3. Jelasity, M., Voulgaris, S., Guerraoui, R., Kermarrec, A.-M., Van Steen, M.: Gossip-based peer sampling. ACM Trans. Comput. Syst. (TOCS) **25**(3), 8 (2007)
4. Khelghatdoust, M.: Gossip based peer sampling in social overlays. Master's thesis, The Royal Institute of Technology (KTH), Stockholm, Sweden (2014)

5. Montresor, A., Jelasity, M.: PeerSim: a scalable P2P simulator. In: Proceedings of the 9th International Conference on Peer-to-Peer (P2P'09), Seattle, WA, pp. 99–100, September 2009
6. Voulgaris, S., Gavidia, D., Van Steen, M.: Cyclon: inexpensive membership management for unstructured p2p overlays. J. Netw. Syst. Manag. **13**(2), 197–217 (2005)