ProFID: Practical Frequent Item Set Discovery in Peer-to-Peer Networks

Emrah Çem[1] and Öznur Özkasap[1]*

Koç University, Istanbul, Turkey

Abstract. This study addresses the problem of discovering frequent items in unstructured P2P networks. We propose a fully distributed Protocol for Frequent Item set Discovery (ProFID) where the result is produced at every peer. We also propose a practical rule for convergence of the algorithm. Finally, we evaluate the efficiency of our approach through an extensive simulation study on PeerSim.

1 Introduction

Peer-to-Peer (P2P) systems are very dynamic and scalable, hence centralized approaches are not as functional, reliable, and robust as decentralized approaches. Peers may need a system-wide information such as network size, query/event counts, or mostly contacted peers for specific files in order to perform various tasks such as load-balancing or topology optimization [1]. Database applications [2], wireless sensor networks [3], and security applications can also make use of frequent item discovery protocol, as well as P2P applications [1, 4]. Hence, efficient discovery of frequent items would be a valuable service for distributed systems.

We propose a fully distributed gossip-based approach named ProFID using pairwise averaging function and convergence rule which is novel in frequent item discovery problem. Our approach for data aggregation in ProFID is inspired by [3] and the main differences are that it works for multiple items and utilizes aggregation for frequent item set discovery. The work of [3] presents a distributed way of calculating aggregates such as averages, sums, and extremal values. They use topology information while determining the termination time, which is not practical since it may not be available at all peers. Another related study [5] proposes a push-synopses protocol using uniform gossip for aggregate computation and analyzes the scalability, reliability and efficiency of their approach. Algorithm converges to true average only if all peers have a knowledge about all items in the system, which might be an inconvenient requirement for large networks without centralized agents. In [4], gossip protocol is used for the first time in frequent item discovery problem. In order to identify frequent elements, a threshold mechanism is used, and by using sampling, the communication load is decreased. However, a uniform gossip is performed, which is not a realistic assumption for large networks.

* Research supported by TUBITAK (The Scientific and Technical Research Council of Turkey) under CAREER Award Grant 104E064.

2 ProFID: Protocol for Frequent Item Set Discovery

We consider a network consisting of N peers denoted as $P=\{P_1, P_2, \ldots, P_N\}$ and M item types denoted as $D=\{D_1, D_2, \ldots, D_j, \ldots, D_M\}$, where D_j has a global frequency g_{D_j}. Parameters *N, M,* and g are system-wide information, hence they are unknown to all peers a priori. Each peer (P_i) has a local set of items $S_i \subseteq D$ and each local item (D_j) has a local frequency f_{i,D_j} such that

$$g_{D_j} = \sum_{i=1}^{N} f_{i,D_j}, \qquad D_j \notin S_i \Longrightarrow f_{i,D_j} = 0$$

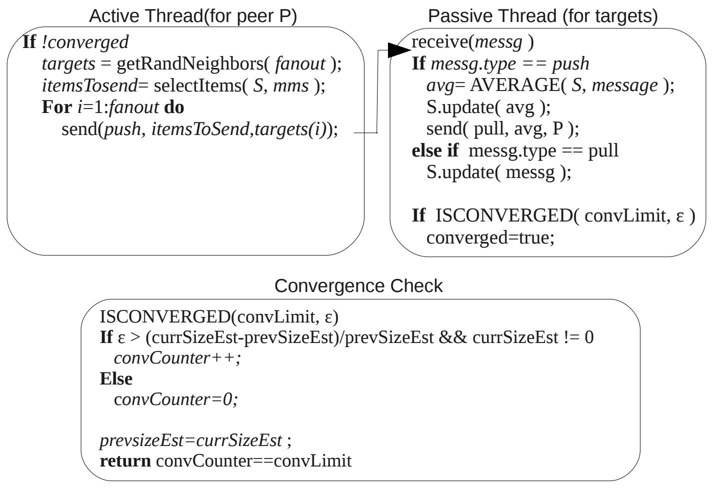
Peers form an unstructured network and communicate in rounds with a fixed duration. A peer may leave or join the network at any time. Furthermore, peers' local clocks do not need to be synchronized because peers use clocks just to perform periodic operations.

We provide a gossip-based fully distributed approach with pairwise averaging function in ProFID, utilizing pairwise averaging function with gossip-based aggregation and a practical convergence rule (Fig. 1). Our pairwise averaging function uses push-pull scheme meaning that a peer sends its state (in a push message) to a target peer and the target peer performs averaging operation using its own state and incoming state, then replies the average of incoming items (in a pull message) back to the sender. Then, sender updates its state. By this way, a single push-pull based pairwise averaging operation is completed. In order to prevent misleading calculations, this operation must be performed atomically. For this purpose, we used buffering and timeout mechanisms. Since we aim to find frequent items, knowing averages of items is not enough, we also need to calculate the system size N at each peer. In order to calculate system size, an initiator peer adds a unique item named *ui* in its local item set. The local frequency of this item is set to 1. Since only one peer has that unique item, average frequency of that item would converge to $\frac{1}{N}$ from which N can be extracted by each peer. Using both estimated average frequency of items and the network size, each peer can calculate the frequencies of items. Due to page limitation, we refer interested reader to [6] for details of our study.

3 Simulation Results

We used PeerSim [7] simulator to build the model for ProFID. We evaluated the behavior and performance of ProFID through extensive large-scale distributed scenarios. Random graphs with average degree 10 is used in the experiments and items are distributed randomly to all peers. Moreover, all the simulation data points are the average of 50 experiments. We evaluate the effects of convergence parameters (ε and *convLimit*) on the accuracy and efficiency of ProFID, as well as the performance of pairwise averaging function.

Fig. 2a depicts the scalability of ProFID in terms of time complexity. Our results (for number of rounds to converge) agree with the O(logN) time complexity of epidemic dissemination [8]. Fig. 2b illustrates that even though the



Active Thread(for peer P)

```
If !converged
    targets = getRandNeighbors( fanout );
    itemsTosend= selectItems( S, mms );
    For i=1:fanout do
        send(push, itemsToSend,targets(i));
```

Passive Thread (for targets)

```
receive(messg )
If messg.type == push
    avg= AVERAGE( S, message );
    S.update( avg );
    send( pull, avg, P );
else if  messg.type == pull
    S.update( messg );

If  ISCONVERGED( convLimit, ε )
    converged=true;
```

Convergence Check

```
ISCONVERGED(convLimit, ε)
If ε > (currSizeEst-prevSizeEst)/prevSizeEst && currSizeEst != 0
    convCounter++;
Else
    convCounter=0;

prevsizeEst=currSizeEst ;
return convCounter==convLimit
```

Fig. 1: ProFID Algorithms: Active thread, passive thread, and convergence check

link drop probability is around 5%, convergence error of the pairwise averaging is almost negligible. In this simulation, the message loss probability of each link is independent and identically distributed. As depicted in Fig. 2c, algorithm converges faster for larger values of fanout since a peer exchanges its state with more neighbors and its state is disseminated faster to the network.

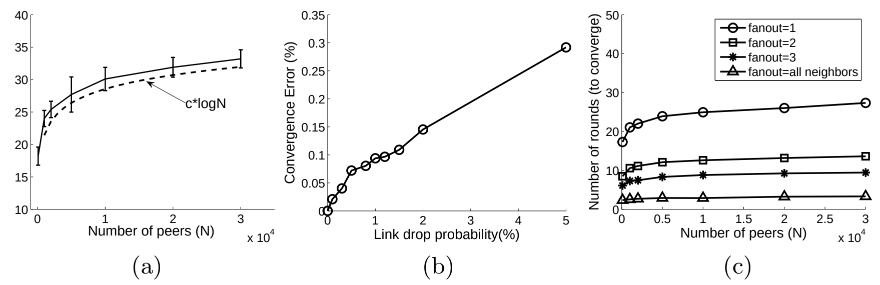
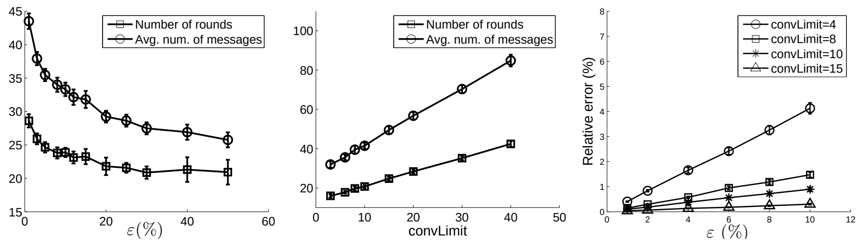


Fig. 2: (a) Number of gossip rounds needed for all peers to converge. (b)The effect of link drop probability on the accuracy of pairwise averaging. (c) The effect of fanout on convergence time

Fig. 3a shows that increasing ε, decreases both average number of messages sent per peer and number of rounds to converge because convergence rule increments *convCounter* value with more probability, which results in faster convergence. Since algorithm converges faster, peers communicate less and average number of messages sent per peer decreases. In contrast to ε parameter, increasing *convLimit* increases both the average number of messages sent per peer and number of rounds to converge because *convCounter* needs to be incremented more to reach *convLimit*. Fig. 3c illustrates the effects of convergence param-



(a) (b) (c)

Fig. 3: The effects of ε (a) and *convLimit* (b) on convergence time and average number of messages sent per peer. (c) The effects of convergence parameters on relative error.

eters on converge time. The fastest convergence occurs whenever ε parameter takes its largest value and *convLimit* takes its smallest value, which agrees with the convergence rule. However, there is a tradeoff between convergence time and accuracy as depicted in Fig. 3c.

In conclusion, our results confirm the practical nature, ease of deployment and efficiency ProFID. As future directions, we aim to evaluate ProFID in peer churn scenarios, and investigate the effect of limited gossip message sizes. For comparison, we are developing the well-known push-synopses protocol [5] by adapting it to the problem of frequent item discovery and practical P2P network settings. Furthermore, we aim to conduct network tests of ProFID on the PlanetLab.

References

1. Li, M., Lee, W.C.: Identifying Frequent Items in P2P Systems. ICDCS. (2008) 36–44
2. Manjhi, A., Shkapenyuk, V., Dhamdhere, K., Olston, C.: Finding (Recently) Frequent Items in Distributed Data Streams. ICDE. (2005) 767–778
3. Jelasity, M., Montresor, A., Babaoglu, O.: Gossip-Based Aggregation in Large Dynamic Network. ACM Ttrans. Comput. Syst. **23** (2005) 219–252
4. Lahiri, B., Tirthapura, S.: Computing Frequent Elements Using Gossip. SIROCCO. (2008) 119–130
5. Kempe, D., Dobra, A., Johannes., G.: Gossip-Based Computation of Aggregate Information. FOCS. (2003) 482–491
6. Çem, E., Özkasap, Ö.: ProFID: Practical Frequent Item Set Discovery in Peer-to-Peer Networks. Technical Report, Koç University, March 2010.
7. The Peersim simulator. http://peersim.sf.net.
8. Özkasap, Ö., Çaglar, M., Yazici, E. S., Küçükçifçi, S.: An analytical framework for self-organizing peer-to-peer anti-entropy algorithms. Perform. Eval. **67** (2010) 141–159