Using Less Links to Improve Fault-Tolerant Aggregation

(Fast Abstract)

Paulo Jesus
pcoj@di.uminho.pt

Carlos Baquero
cbm@di.uminho.pt

Paulo Sérgio Almeida
psa@di.uminho.pt

Universidade do Minho, Departamento de Informática (CCTC/DI)

I. Motivation

Data aggregation[1] plays a basal role in the design of scalable distributed applications [1], allowing the determination of meaningful system-wide properties to direct the execution of the system. For instance, aggregation can be used to estimate: the size of the network to dimension of Distributed Hash Table (DHT) structures [2], or to set a quorum in dynamic settings [3]; the average system load to guide local load-balancing decisions; the total network disk space in a P2P sharing system. In the particular case of Wireless Sensor Networks (WSN), due to energy constraints, data collection is often only practicable if aggregation is performed.

Several aggregation algorithms have been proposed in the recent years, tackling the problem for different settings, and yielding different characteristics in terms of accuracy, time and communication tradeoffs. Traditional tree-based approaches [4], [5], [6] rely on the existence of a hierarchic aggregation structure (e.g. spanning tree) to perform in-network aggregation, but a single point of failure can jeopardize the obtained result. Common gossip aggregation algorithms [7], [8], [9], [10] are based on the execution of iterative *averaging* techniques for all nodes to converge to the correct result. Nonetheless, the correctness of such algorithms depends on a fundamental invariant commonly designated as "mass conservation", which is broken by node crashes and message losses, unpredictably affecting the estimated results [11], [12].

The majority of existing aggregation techniques are found lacking in terms of fault-tolerance, being unable to simultaneously provide accuracy and efficiently tolerate faults. To the best of our knowledge, only a recent approach has successfully tackled this issue: *Flow Updating* [12]. *Flow Updating* is an aggregation algorithm that provides an accurate estimate (converges to the correct value) at all nodes, and it is by design immune to message loss. We propose a simple heuristic that extends the initial version of *Flow Updating* by locally ignoring some communication links between neighbors while keeping it's fault-tolerance properties. We also provide some preliminary results and discuss the benefits of the proposed extension.

[1]We refer to data aggregation as the distributed computation of aggregation functions (e.g. COUNT, AVERAGE, SUM).

II. An Extension to Flow Updating

Flow Updating is inspired on existing gossip-based aggregation approaches, but unlike them it tolerates faults by design. The algorithm is based on the concept of *flow* (from graph theory). In a nutshell, an averaging iterative process is executed at each node, keeping the initial input value unchanged and performing idempotent flow updates. At each round, nodes compute the average of known estimates (received from neighbors) and update the neighbors flows in order to converge to it. The new computed estimate and the flows are sent in a single message (local broadcast) to all neighbors. The estimation of the aggregation result can be produced at each node from it's initial value and the flow to each neighbor. More details about the algorithm can be found in [12].

A. Extension

We observe that during the execution of *Flow Updating*, a node will commonly receive data targeted to other nodes that will not be used, since nodes locally broadcast data for all known neighbors in a single message. Furthermore, if two nodes share common neighbors they may receive from different paths (neighbors) data that has already taken into account the estimate of the other node, ending up multiplying its influence in the next averaged estimate. For instance, considering three nodes i, j, and k directly connected to each other, the data (estimate and flow) targeted to i from j may already consider the estimate of k, and the one received by i from k may also take into account the estimate of j, duplicating the influence of j and k in the next estimate computed by i. In this situation, node i will only need to receive the message from one of those neighbors, since they already take into consideration the estimation of each other. Attending to this, we propose an extension to *Flow Updating* in order to reduce the communication required to perform aggregation and improve it's performance.

The main idea is to reduce the exchange of redundant data between nodes that share the same neighbors, by breaking local communication cycles. To achieve this, we defined a simple heuristic that takes advantage of all neighbors data contained in each message, to find shared neighbors and locally decide to ignore some communication links between them. In particular, upon receiving a message from neighbor j,

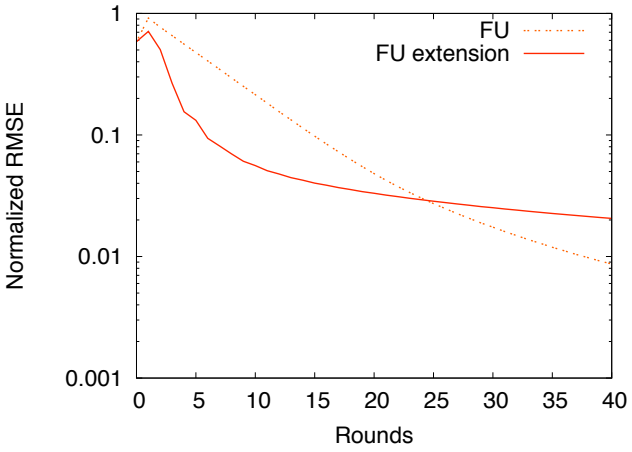

Fig. 1. Convergence speed of Flow Updating Vs Flow Updating Extension (2D/mesh networks, $n = 1000$, and $d \approx 10$).

node i scans it's contents looking for neighbors in common. If a shared neighbor k is found, one of the communication links between them will be "deactivated" according to a predefined criteria. We simply rely on node IDs so that each node independently decides which link will be deactivated (e.g. link between the two nodes with the greater IDs). The execution of *Flow Updating* proceeds normally, only ignoring deactivated neighbors at each node (e.g. estimate and flow removed from the local state).

III. Evaluation

We obtained some preliminary results by comparing *Flow Updating* with the proposed extension in a synchronous simulation environment. In particular, we considered a network in which communication links between nodes are established according to their geographical proximity (2D/mesh). Both algorithms are used to determine the AVERAGE of different input values distributed uniformly at random across the network. Figure 1 shows the average convergence speed (number of rounds to reach a given accuracy) from 50 repetitions of the execution of the compared algorithms under identical settings, generating different 2D/mesh networks with the same size ($n = 1000$) and average connection degree ($d \approx 10$) in each repetition.

Results show that the *Flow Updating* extension outperforms the basic version of the algorithm in terms of convergence speed at an early stage, requiring approximately half of the number of rounds to reach an average accuracy of 10% across the network (normalized RMSE $= 0.1$), but the speed of the extended version progressively slows down and it is surpassed by *Flow Updating* for accuracies $< 3\%$ (from the 25th round). Furthermore, *Flow Updating* extension allowed the average deactivation of approximately 75% of the existing links in the considered scenario. This preliminary results turned out to be very interesting and promising, since the proposed heuristic allowed the seamless (relying only on data available in *Flow Updating*, without any additional mechanism) definition of a "lighter" communication overlay (using only a quarter of the initial communication links) that increases the convergence speed of the algorithm.

IV. Conclusion

In this work, we extend *Flow Updating* [12] – a fault-tolerant aggregation algorithm, by deactivating some communication links between nodes that share a common neighborhood. A simple heuristic is proposed, relying only on the data available from the algorithm to enable each node to independently decide which links can be deactivated.

Some empirical results are provided, showing that the proposed heuristic initially improves the performance of the aggregation process, and also considerably reduces the number of active communication links. The obtained results are interesting and promising. A thorough and wider study of the effect of link remotion on aggregation algorithms is required in order to fully understand the potential of this research direction.

References

[1] R. V. Renesse, "The importance of aggregation," *Future Directions in Distributed Computing, Lecture Notes in Computer Science*, vol. 2584, pp. 87–92, 2003.

[2] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 149–160, Aug 2001.

[3] I. Abraham and D. Malkhi, "Probabilistic quorums for dynamic systems," *Distributed Computing, Springer Berlin/Heidelberg*, vol. 18, no. 2, pp. 113–124, Dec 2005.

[4] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "TAG: a Tiny AGgregation service for ad-hoc sensor networks," *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 131–146, Dec 2002.

[5] J. Li, K. Sollins, and D. Lim, "Implementing aggregation and broadcast over distributed hash tables," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 1, pp. 81–92, 2005.

[6] Y. Birk, I. Keidar, L. Liss, A. Schuster, and R. Wolff, "Veracity radius: capturing the locality of distributed computations," *PODC '06: Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, Jul 2006.

[7] D. Kempe, A. Dobra, and J. Gehrke, *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*, pp. 482– 491, 2003.

[8] M. Jelasity, A. Montresor, and O. Babaoglu, "Gossip-based aggregation in large dynamic networks," *ACM Transactions on Computer Systems (TOCS)*, 2005.

[9] J.-Y. Chen, G. Pandurangan, and D. Xu;, "Robust computation of aggregates in wireless sensor networks: Distributed randomized algorithms and analysis," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 9, pp. 987 – 1000, Sep 2006.

[10] F. Wuhib, M. Dam, R. Stadler, and A. Clemm, "Robust monitoring of network-wide aggregates through gossiping," *10th IFIP/IEEE International Symposium on Integrated Network Management*, pp. 226 – 235, 2007.

[11] P. Jesus, C. Baquero, and P. S. Almeida, "Dependability in aggregation by averaging," in *Simpósio de Informática (INForum)*, September 2009 (in press).

[12] P. Jesus, C. Baquero, and P. S. Almeida, "Fault-tolerant aggregation by flow updating," in *9th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS)*, ser. Springer LNCS, vol. 5523, Lisbon, Portugal, June 2009, pp. 73–86.