Aggregation Protocols in Light of Reliable Communication

Ziad Kassam*, Ali Shoker*, Paulo Sérgio Almeida† and Carlos Baquero†
HASLab, INESC TEC & Minho University, Portugal
*{zakassam,ali.shoker}@inesctec.pt
†{psa,cbm}@di.uminho.pt

***Abstract*—Aggregation protocols allow for distributed lightweight computations deployed on ad-hoc networks in a peer-to-peer fashion. Due to reliance on wireless technology, the communication medium is often hostile which makes such protocols susceptible to correctness and performance issues. In this paper, we study the behavior of aggregation protocols when subject to communication failures: message loss, duplication, and network partitions. We show that resolving communication failures at the communication layer, through a simple reliable communication layer, reduces the overhead of using alternative fault tolerance techniques at upper layers, and also preserves the original accuracy and simplicity of protocols. The empirical study we drive shows that tradeoffs exist across various aggregation protocols, and there is no one-size-fits-all protocol.**

***Index Terms*—Distributed Systems; aggregation protocols; Push-Sum; Flow Updating; performance.**

1. Introduction

With the prominence of the Internet of Things (IoT), Mobile Computing (MC), and Wireless Sensor Networks (WSN), data aggregation is becoming a de-facto technique to manipulate distributed data, control the network, and perform lightweight computations (like max, sum, count…) [4], [7], [11]. In such systems, resource constrained devices are usually deployed over ad-hoc networks, mainly using wireless communication media (WIFI, Bluetooth, Zig-bee), in a Peer-to-Peer (P2P) fashion to avoid the usual bottlenecks of centralized solutions and communication failures. Consequently, the underlying communication layer is often hostile and can compromise the correctness and performance of aggregation protocols; therefore, fault tolerance techniques are still being advocated to make these protocols more reliable and practical [2]–[4], [7], [8].

Specifically, convergence — to a common value across nodes — is considered the prime challenging correctness measure of aggregation protocols since it can easily be adapted to abstract other computations (like sum, count, max, etc.) [8], [11]. Convergence becomes more challenging when subject to environmental impacting factors like the number of nodes in the system, the topology of the network, the reliability of communication medium, the available resources of devices, etc. This leads to several tradeoffs between simplicity, correctness, and performance. Among these protocols, gossip-based aggregation protocols are often considered more robust under such factors [10], [13].

In particular, there are two categories of gossip-based aggregation protocols in literature. The first, is simple, in which an (mutable) estimate of the average value is computed locally and then propagated to other nodes either in a completely distributed way, e.g., Push-Sum [11], or in a clustered way as in DRG [4]. This simplicity however comes at the price of dedicated fault tolerance techniques, required to handle message loss and duplication, whose solutions are sometimes costly [5]. Another category, like Flow-Updating (FU) [8], is immune to message loss and duplication by nature due to the concept of flows: idempotent averaging estimates are locally computed based on average flows received from other nodes in an immutable way. As the authors show in [8], [9], FU is not prone to transient communication failures, but as we show later, it exhibits some instability under long-lived communication failure and when the average degree (i.e., number of neighbors of a node) is high. To take advantage of idempotency, several "hybrid" protocols [6], [14] tried to integrate the idea of flows and node backups with the Push-Sum protocol in an attempt to introduce a simple and fault tolerant aggregation protocol; unfortunately, this yielded other accuracy and performance issues [13]: (1) performance was significantly impacted by communication issues, and (2) some accuracy in convergence is detected.

In this paper, we present an empirical study that compares the above variants of gossip-based protocols leading to the following conclusion: once simple classical protocols, like Push-Sum [11], are supported by a robust exactly-once underlying communication layer, they may outperform other protocols [6], [8], [14], and importantly, maintain the original convergence accuracy.

The rest of the paper is organized as follows. Section 2 gives a quick background on the addressed categories of protocols, and Section 3 conveys the empirical results followed by concluding remarks.

The second author is supported by SMILES track of TEC4Growth project (NORTE-01-0145-FEDER-000020), and the third and forth authors are supported by EU H2020 LightKone project (732505).

978-1-5386-1465-5/17/$31.00 © 2017 IEEE

2. Background and Related Works

In this section, we present a concise background on three variants of aggregation protocols: Push-Sum (PS), Flow Update (FU), and Distributed Random Grouping (DRG). Despite the diverse data aggregation protocols introduced in literature [4], [6], [12], [14], we opt for the aforementioned protocols being well-known and most other protocols are variants using their core concepts.

Push-sum protocol [11] is a simple gossip-based protocol where each node divides its local value by half and propagates it to other peers until convergence is achieved. PS protocol converges faster when degree increases, however the algorithm correctness relies on "mass" conservation [11] where any kind of system failure violates mass conservation. Consequently, several variants like [5], [6], [14] were introduced to withstand network and node failures, which resulted in accuracy and performance overheads [13].

Flow-Update (FU) [8], is based on the concept of idempotent computation through "flows". The idea is that each node calculates the average based on the all contributions of the in/out flows along the edges of the neighbors and its initial value. Since this depends on the direct flows (and the initial intact value), there is no need to retain corresponding mutable variables. This makes the algorithm natively tolerant to message loss and duplication, but suffers some instability period upon recovery from network partitions (as flows are missing).

LiMoSense [5], Push-Flow [6] and Push-Cancel-Flow [14] followed a hybrid model by using concepts from PS and FU through using flows for data exchange and mutable local histories to compute the estimates. However, these protocols induced correctness problems as accuracy and division by zero [10], [14].

DRG [4] is an aggregation protocol that essentially consists in the continuous random creation of groups across the network, in which messages are broadcast and aggregates are successively computed (averaged). In DRG, message loss between coordinators and neighbors may happen; and thus, partial fixes to avoid deadlock of nodes waiting forever may result in violating mass conservation [8].

3. Evaluation

3.1. Experimental Setup

To perform our experiments, we used a locally developed simulator that runs on a single machine, where message loss/duplication, network partition, network topology, number of nodes and the degree can be customized to serve our purpose. To assess the protocols under reliable communication, we tried to use an exactly-once protocol, inspired from [1], whose messaging abstraction allows us to experiment the system without using a real communication layer (e.g., IP protocol). The experiments considered four network topologies with random generation of links: Bus, Ring, 2D Mesh, and random graph of several dimensions, i.e., different degrees (from 3 to 20). However, the paper only presents those of random graph since it allows for a wide range of degrees, and more realistically represents wireless settings.

We consider the protocols PS, FU and DRG protocols being well-known or represent the state-of-the-art of gossip-based aggregation protocols. FTPS and FTDRG refer to the fault-tolerant versions of PS and FU where a reliable communication layer is integrated.

Finally, the protocols are evaluated through two main metrics: accuracy and speed, considering message loss and duplication under different graph degrees. Accuracy is expressed by the normalized Root Mean Square Error (RMSE) of the estimate in contrast to the target value, and the speed expressed by the number of iterations to reach this accuracy. An estimation of the messaging overhead with and without the reliable communication plug-in is also presented at the end. For the sake of completeness, we tried to implement and experiment the Push-Flow protocol, and we noticed that the accuracy is significantly lost with 1000 nodes. Indeed, this result is consistent with those in [13] that shows the accuracy problem starting with 60 nodes up.

3.2. Convergence speed in a fault-free network

Convergence speed is experimented as the degree increases from 3 to 20, using random graph of 1000 nodes. We present the number of iterations required by each protocol to achieve convergence in a fault-free scenario, where convergence is considered achieved once the $RMSE = 10^{-11}$.

Figure 1 interestingly shows that the convergence speed of FU improves until reaching degree 7, beyond which more iterations are needed to converge. The results look surprising for the first glance, however, they are consistent with the results in the original paper of FU [8] for degree 10. This behavior is referred to the direct dependence of FU on the in/out flows of direct neighbors in calculating the $estimate$ of the average. As the degree increases, the number of flows per node increases, thus significantly modifying the $estimate$. Indeed, this conforms with the analysis in [8] that shows FU under link failures converges faster than healthy network as the number of links drops.

To the contrary, PS and DRG converge much faster (log-scale is shown) with higher degrees and they significantly outperform FU starting at degree 7 and 12, respectively. (Notice that PS and DRG are confounded with FTPS and FTDRG, respectively, given that no faults occur.) This behavior is expected as the $estimate$ in DRG and PS is computed and spread to neighboring nodes, thus as the network is more connected, the information propagates faster.

Finally, the figure shows that PS converges faster than DRG, especially, when the degree is low since DRG cannot create large groups in this case, and this broadcast to all system nodes takes longer, contrary to large groups (with large degree).

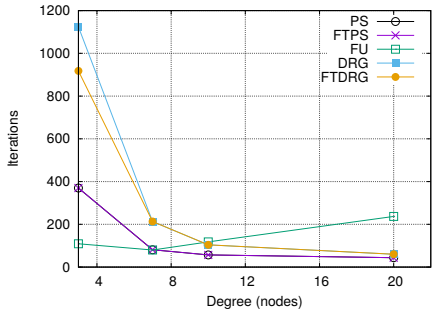

Figure 1. Convergence speed in fault-free scenario.

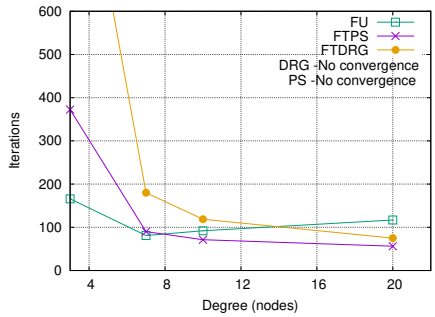

Figure 2. Estimated convergence speed under 20% message loss.

3.3. Convergence speed under loss and duplication

Under message loss, the above behavior changes as shown in Fig 2. We performed experiments using different percentages of message loss, i.e., from 10% to 50%, and noticed that the patterns are very similar, and thus we only convey the 20% results in Fig. 2. The convergence speed of FU under message loss remains equivalent as in the fault-free scenario (Fig. 1), or even better, which is also demonstrated in [8], where the number of flows is small and the computation of the estimate depends directly on it. Thus, no need to integrate the reliable communication layer with FU. In contrast, PS and DRG cannot converge at all due to violating mass conservation.

Using the fault tolerant communication layer, the variants of PS and DRG, i.e., FTPS and FTDRG, are again able to operate, however at an additional communication overhead. Since the communication layer is transparent to the protocols' logic, this overhead does not manifest on the number of iterations needed. Counting an additional iteration per retransmitted message is unfair, since not the entire system is actually delayed, whereas discarding the retransmission time and overhead is biased to PS. To handle this, we estimated the number of extra iterations needed through dividing the extra messages propagated due to failures, by the *average number of messages exchanged per iteration* in the fault-free case. This estimation turned out to be a polynomial equation of the form: $f(x) = \Sigma_{i=1}^{4} x^i$; where x is the retransmission ratio. In the case of 20%, $f(0.2) \approx 0.25$. Using this estimation, the convergence speed

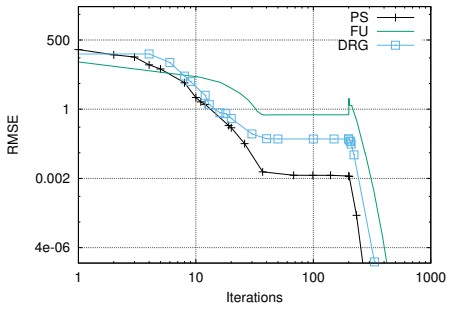

Figure 3. Root Mean Square Error under 10% network partition size and degree 10.

of FTPS and FTDRG is roughly 25% lower than that of PS and DRG in the fault-free case (Fig. 1), respectively. Therefore, FTPS only overtakes FU starting from degree 10 (instead of 7 in the fault-free case). The overhead is higher for FTDRG which overtakes FU starting from degree 16 and up, and thus we believe it is not worth using DRG instead of FU in this case.

As for message duplication, FU and DRG are not affected since computing the aggregate is idempotent; and therefore, the convergence speed remains as shown in Fig. 1. To the contrary, PS suffers from message duplication and thus cannot converge at all — thus we do not plot the corresponding curve. The use of the fault tolerant communication layer in FTPS overcomes the duplication problems as expected without significant changes in convergence speed to this presented in Fig 1.

3.4. The impact of network partitions

To experiment the protocols under network partitions, we used a random network of 1000 nodes, and manually identified some links that when broken can lead to network partition. Following this process, we have studied the time needed by the three protocols PS, FU, and DRG to converge under different settings: changing the partition size (to 10%, 25%, and 50%) and time of partitioning occurs (i.e., 20-200 iterations and 200-500 iterations) with degree $\approx$ 10. In particular, we studied the time needed to converge to a very small RMSE.

Due to size limits, we only convey the important part of our results in Figure 3 which correspond to 10% partition size and partition time 20-200 iterations. This graph is chosen due to the following reasons. Changing the partition size to 25% and 50% lead to very similar patterns to those in Figure 3 with a slight difference that the impact of partitioning is a bit lower on the two partitions. This reason is referred to the fact that initial values are quickly propagated to most of the nodes in the two partitions, leading to faster averaging before partitioning occurs. This very reason lead us to omit the experiments of partitioning time 200-500 iterations. Indeed, the three protocols reached an almost stable state after 200 iterations which absorbs the impact of partitioning.

TABLE 1. ESTIMATED MESSAGE'S HEADER SIZE OF FU, DRG, PS, AND TWO FAULT TOLERANT PIGGY-BACKING PATTERNS OF PS.

FU	DRG	PS	FTPSPB1	FTPSPB2
76	36 (max)	74	32	92

Considering Figure 3, the first observation is that, at the instant partitioning starts (i.e., 20 iterations), the two partitions continue to converge through tens of iterations before stabilizing almost at 100 iterations in all protocols. This was expected as the three protocols are completely decentralized and can operate as long as peers are reachable. The second observation shows that FU shows some fluctuation when the network heals from partition as shown by the spike at the iteration 200 in Figure 3. To further understand this behavior, we tried to compute the value that each partition converged to before and after healing. Our experiments show that FU tried to converge to different values on each partition which is far from the optimal average. To the contrary, the values of PS and DRG keep getting closer to the optimal average, though slowly. This is referred to the high impact of using immutable flows in FU, which is consistent with the results in [14], and which is considered the major drawback in flow based protocols.

3.5. The messaging overhead

As for message's header size, Table 1 presents the average size of the three protocols FU, PS, and DRG: 76, 74, and 36, respectively. Despite this slight difference among protocols, it has no impact in a real network. In fact, given that the payload in aggregation applications is often small (e.g., few Bytes), the header and the payload can both fit in a single UDP datagram. (TCP is not recommended in such hostile settings.) To tolerate message loss and duplication, FTPS should also be considered. According to the exactly-once communication protocol we used [1], FTPS requires four different message types, in two round-trip delays, to deliver a single aggregation message and a corresponding ACK. The size of each message header ranges between 16 and 74 Bytes. However, under congestion, messages can be piggy-backed in two patterns depicted as FTPSPB1 and FTPSPB2 in Table 1. Therefore, the messaging overhead of the exactly-once layer is negligible as well.

4. Learned Lessons and Conclusions

Complementary to state-of-the-art studies on aggregation protocols, we focused in our study on taking failure prone protocols as PS and DRG, and integrate a reliable communication layer that can preserve the simplicity of the classical protocols to overcome mass conservation risks.

Our conclusion is that Flow-Updating protocol [8] is natively robust to loss and duplication but not to network partitions which incur some temporary perturbation in the values. Since network partitions are more likely to occur once the degree is small (e.g., 3), it is recommended to avoid using FU unless the perturbation in the values is tolerable by the application; the alternative is to use other Flow-Updating variants like [6], [14] only if high accuracy is not a matter. On the other hand, PS and DRG are prone to communication issues, and thus using a robust communication layer is crucial for mass conservation. The experiments we conveyed show that providing a reliable communication layer comes at a cost, which is not worth it in the case of DRG. To the contrary, PS protocol is more accurate and outperforms the other protocols when the $degree > 10$ despite the overhead of using a reliable communication layer.

In our simulations, and previous works as well, other network problems like congestion and interference were not considered, which we believe are worth considering if a real experimentation environment is available.

References

[1] P. S. Almeida, A. Shoker, C. B. Moreno, A. Shoker, P. S. Almeida, and C. B. Moreno. Exactly-once quantity transfer. 2015.

[2] C. Baquero, P. S. Almeida, and R. Menezes. Fast estimation of aggregates in unstructured networks. In *Autonomic and Autonomous Systems, 2009. ICAS'09. Fifth International Conference on*, pages 88–93. IEEE, 2009.

[3] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Gossip algorithms: Design, analysis and applications. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pages 1653–1664. IEEE, 2005.

[4] 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*, 17(9):987–1000, 2006.

[5] I. Eyal, I. Keidar, and R. Rom. Limosense: live monitoring in dynamic sensor networks. *Distributed computing*, 27(5):313–328, 2014.

[6] W. N. Gansterer, G. Niederbrucker, H. Straková, and S. Schulze Grotthoff. Robust distributed orthogonalization based on randomized aggregation. In *Proceedings of the second workshop on Scalable algorithms for large-scale systems*, pages 7–10. ACM, 2011.

[7] M. Jelasity, A. Montresor, and O. Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Transactions on Computer Systems (TOCS)*, 23(3):219–252, 2005.

[8] P. Jesus, C. Baquero, and P. S. Almeida. Fault-tolerant aggregation by flow updating. In *DAIS*, pages 73–86. Springer, 2009.

[9] P. Jesus, C. Baquero, and P. S. Almeida. Flow updating: Fault-tolerant aggregation for dynamic networks. *Journal of Parallel and Distributed Computing*, 78:53–64, 2015.

[10] P. Jesus, C. Baquero, and P. S. Almeida. A survey of distributed data aggregation algorithms. *IEEE Communications Surveys & Tutorials*, 17(1):381–404, 2015.

[11] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*, pages 482–491. IEEE, 2003.

[12] D. Liu and M. Prabhakaran. On randomized broadcasting and gossiping in radio networks. *Computing and Combinatorics*, pages 643–654, 2002.

[13] G. Niederbrucker and W. N. Gansterer. Robust gossip-based aggregation: A practical point of view. In *2013 Proceedings of the Fifteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 133–147. SIAM, 2013.

[14] G. Niederbrucker, H. Straková, and W. N. Gansterer. Improving fault tolerance and accuracy of a distributed reduction algorithm. In *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:*, pages 643–651. IEEE, 2012.