Symmetric Push-Sum Protocol for Decentralised Aggregation

Francesco Blasa, Simone Cafiero, Giancarlo Fortino,
Department of Electronics, Informatics and Systems
University of Calabria, 87036 Rende (CS), Italy
{checco84,simone.cafiero}@gmail.com, g.fortino@unical.it

Giuseppe Di Fatta
School of Systems Engineering, The University of Reading
Whiteknights, Reading, Berkshire, RG6 6AY, UK
G.DiFatta@reading.ac.uk

***Abstract*—Gossip (or Epidemic) protocols have emerged as a communication and computation paradigm for large-scale networked systems. These protocols are based on randomised communication, which provides probabilistic guarantees on convergence speed and accuracy. They also provide robustness, scalability, computational and communication efficiency and high stability under disruption. This work presents a novel Gossip protocol named Symmetric Push-Sum Protocol for the computation of global aggregates (e.g., average) in decentralised and asynchronous systems. The proposed approach combines the simplicity of the push-based approach and the efficiency of the push-pull schemes. The push-pull schemes cannot be directly employed in asynchronous systems as they require synchronous paired communication operations to guarantee their accuracy. Although push schemes guarantee accuracy even with asynchronous communication, they suffer from a slower and unstable convergence. Symmetric Push-Sum Protocol does not require synchronous communication and achieves a convergence speed similar to the push-pull schemes, while keeping the accuracy stability of the push scheme. In the experimental analysis, we focus on computing the global average as an important class of node aggregation problems. The results have confirmed that the proposed method inherits the advantages of both other schemes and outperforms well-known state of the art protocols for decentralized Gossip-based aggregation.**

***Keywords*-peer-to-peer computing; distributed aggregation algorithms; gossip protocols; extreme scale computing.**

I. Introduction

Nowadays, highly distributed systems such as P2P networks, large scale sensor networks, grids and ubiquitous systems, enable a broad range of applications [1]. Centralized paradigms are not suitable for distributed large-scale scenarios as they introduce bottlenecks and failure intolerance. In particular, applications require atop such systems a protocol layer that can cope well with the highly dynamic and decentralized nature of these infrastructures. Locality has been a key element to successfully deploy applications into large-scale networked systems. However, computing and spreading global information is still necessary for a wide range of applications and is a particularly challenging task when considering dynamic, highly distributed, large and extreme scale systems.

Aggregation protocols represent a decentralized paradigm for computing global properties of distributed systems. Several distributed aggregation protocols have been proposed in the last years. They can be divided into two main classes: Tree-based protocols and Gossip-based protocols.

The former performs a tree-based communication throughout a tree overlay structure (e.g., [2]–[4]). Tree-based protocols support a minimum number of communications but require the construction of a hierarchical communication structure among nodes and can be affected by the presence of single points of failure.

The second class includes Gossip (or Epidemic) protocols, which are a robust and scalable communication paradigm to disseminate information in a large-scale distributed environment using randomised communication [5], [6]. Although Epidemic protocols have communication costs usually greater than tree-based protocols, they are intrinsically fault tolerant. Gossip-based communication can use push, pull or push-pull schemes.

P2P applications based on Gossip protocols are emerging in many fields. In [7], a global load monitoring service for P2P overlay networks has been proposed. In [8], a decentralized dynamic load balancing algorithm for a desktop Grid environment is presented. The work in [9] introduces an epidemic content search mechanism in unstructured P2P overlay over intermittently connected mobile ad hoc networks. The work in [10] studies Gossip-based message dissemination schemes to be employed for content and service dissemination or discovery in unstructured P2P and ad hoc networks. In [11], authors define a protocol to achieve mutual anonymity in unstructured P2P networks, which deals with high churn rates by means of an epidemic-style data dissemination. A number of Gossip-based protocols for sensor networks have also been proposed (e.g., [12]–[14]). Gossip protocols have also been adopted to solve the general data aggregation problem [15]–[17].

In this work, we present a new algorithm for Gossip-based aggregation named Symmetric Push-Sum Protocol (SPSP). The proposed algorithm is fully decentralized and suitable for large scale networks. We evaluate performances of our algorithm w.r.t. Push-Sum Protocol (PSP) [15] and Push-Pull Gossip Protocol (PPG) [16], [17]. SPSP preserves the mass conservation invariant, i.e., at any time the sum of all

This work was partly carried out while Francesco Blasa and Simone Cafiero were at the University of Reading, UK, for a work placement of the Erasmus Training Programme (June-Sept. 2010).

values in the network is constant. This invariant guarantees the correctness of aggregation algorithms [18]. In particular, among the various aggregation functions in the experimental analysis we focus on the average. The simulations have confirmed the quality and the consistency of the results. In particular, the results show that the proposed approach always performs much better than the state of the art aggregation protocols for large-scale distributed systems.

The rest of this paper is organized as follow. Section II reviews related work. Section III presents the proposed aggregation protocol. Section IV presents an experimental comparative analysis. Finally, Section V provides some conclusive remarks and future research directions.

II. Related Work

Several Gossip-based aggregation protocols have been proposed. In [15], a push scheme protocol named Push-Sum Protocol (PSP) is proposed. It is a simple aggregation protocol for computing several aggregation functions (e.g., sum, average, count). In PSP, the local scalar value is represented as a pair (v, w), where v is initialised with the local value x and the initial weight w depends on the global aggregate function to be computed as shown in Table I. The global aggregate value is given by v/w after a fixed number of communication cycles. At each cycle, each node halves its local value and weight $(v, w) = (v/2, w/2)$ and sends the new obtained pair to a randomly selected node according to a uniform probability density function (pdf). The global mass is guaranteed to be conserved in case a reliable communication protocol is used. A number of messages equal to the number of nodes in the network is sent in total at each cycle. The *diffusion speed* is the minimum number of protocol cycles required to achieve a good approximation of the true value of the global aggregate function with high probability:

$$Prob(e_i < \varepsilon) \geq 1 - \delta, \quad \forall\, i = 1, \ldots, n, \tag{1}$$

where n is the network size, e_i the approximation error at node i and ε and δ two arbitrary small positive constants. The *diffusion speed* of the PSP has been shown to have a complexity $O(\log(n) + \log(\varepsilon^{-1}) + \log(\delta^{-1}))$ [15].

In [19], authors discuss some issues of PSP when used as summarisation algorithm. The sum function in [15] requires a leader election; this represents a non trivial task and could introduce single points of failure. In [19], the authors provides a scalable and fault tolerant solution to this problem by incorporating a leader election mechanism in the aggregation protocol.

In [16], [17], two similar algorithms are proposed; they can be both referred to as Push-Pull Gossip protocol (PPG). PPG uses a push-pull scheme that improves the diffusion speed w.r.t. PSP. In PPG at each cycle a node i randomly chooses a node j to perform an averaging operation and to update their local values $\frac{x_i + x_j}{2}$. In PPG, $2 \times n$ messages are sent in total at each cycle. In [17], authors focus on the design of Gossip algorithm by defining a method to obtain the fastest Gossip algorithm over a given distributed network. In particular they find out that the averaging time (which is directly related to diffusion speed) depends on the eigenvalue of a doubly stochastic matrix characterizing the algorithm. The fastest Gossip algorithm is obtained by minimizing the eigenvalues in a distributed fashion.

In [18], authors state that the correctness of such algorithms depends on the mass conservation invariant. They show that PPG could violate this fundamental invariant if an *atomic violation* happens. An *atomic violation* occurs on node i when i receives a push while it is waiting for a pull. Therefore two versions of PPG are proposed: Push-Pull Back Cancellation and Push-Pull Ordered Wait. The first algorithm adopts a simple message cancellation mechanism to guarantee atomicity and avoid mass conservation violation. However, the cancellation method decreases the diffusion speed. The second approach adopts a buffer for storing push messages that are received while a push-pull operation is being executed. This mechanism could introduce a deadlock across the network. To avoid deadlocks they introduce a total order among nodes and a constraint in the nodes selection mechanism, which penalizes selection of some nodes.

In general, the use of synchronous cycles simplifies the analysis and the implementation of Gossip-based aggregation protocols. Nevertheless, the protocols can be implemented in completely asynchronous environments. Independent local Poisson clocks can be used to generate synchronous cycles in asynchronous distributed environments (e.g., [20]). A second interesting alternative is the use of an exact global estimation of the right termination time, similar to the median-counter algorithm [6] for rumour spreading.

III. Symmetric PSP

We propose a novel Gossip-based aggregation protocol, the Symmetric Push-Sum Protocol (SPSP), which combines the simplicity of PSP and the convergence speed of PPG.

We assume that the transport protocol is reliable. This assumption is not strictly necessary and could be relaxed as Gossip protocols are intrinsically fault tolerant. However, in this work the effect of packet loss is not investigated.

SPSP adopts an asynchronous push-pull communication scheme. Push-pull schemes are expected to converge faster than push schemes with the same number of exchanged messages [6].

Let consider a distributed system composed by n peers $P = \{P_1, \ldots, P_n\}$. Each node i holds a local value v_i and a local weight w_i ($w_i \geq 0$) and needs to compute a global aggregation function $F(v_1, w_1, \ldots, v_n, w_n)$. Similarly to PSP, SPSP can perform several aggregation functions, some of which are shown in Table I.

Table I
SETTINGS FOR SEVERAL AGGREGATION FUNCTIONS

Function	Description
Sum	$v_i =$ local value $w_i = 1$ at a single node, 0 at all other nodes
Count	$v_i = 1$ $w_i = 1$ at a single node, 0 at all other nodes
Average	$v_i =$ local value $w_i = 1$
Weighted Average	$v_i =$ local value $\times$ local weight $w_i =$ local weight

```
At each node $i$
Require: $v_0, w_0$
    The initial local value, $v_0$;
    The initial local weight, $w_0$.
Initialisation:
1: $(v, w) = (v_0, w_0)$
At each cycle:
2: $j \leftarrow$ getNode()
3: $v \leftarrow v/2$, $w \leftarrow w/2$
4: send an aggregation message to $j$, $\langle (v, w), \textbf{true} \rangle$
At event: received an aggregation message $\langle (v', w'), r \rangle$
    from $j$
5: if $r$ is true then
6:     $v \leftarrow v/2$, $w \leftarrow w/2$
7:     send an aggregation message to $j$, $\langle (v, w), \textbf{false} \rangle$
8: end if
9: $v \leftarrow v + v'$, $w \leftarrow v + w'$
```

Figure 1. The pseudocode of the Symmetric Push-Sum Protocol

As shown in Figure 1, at each cycle a node i randomly selects a communication partner j according to a uniform pdf. This selection is provided by the service *getNode()* (line 2). Then i halves its local value and weight (line 3) and sends them to j (line 4). At the reception of the message, the node j will asynchronously perform a symmetric push operation: it halves its local value and weight and sends them to node i (lines 6-7). Then, it adds the received value and weight to its own value and weight (line 9). In case of *atomic violation*, a node i receives the symmetric push from $k \neq j$ immediately after its push operation, the symmetric push mechanism guarantees the *mass invariant*.

At each random push an asynchronous reply follows as a symmetric push: at each cycle $2 \times n$ messages are sent.

A. Node Cache Protocol

In uniform Gossip protocols the random node selection is a critical operation. In general, the global network topology is not known or is not available at each node.

Figure 2 describes the node selection algorithm adopted in SPSP, i.e. the Node Cache Protocol. The protocol only requires a few assumptions: the network is a connected graph, each node knows its physical neighbours ($Neighbours$), a multi-hop routing protocol is available.

```
At each node $i$
Require: $Q_{MAX}, Neighbours$
    The maximum size of local cache, $Q_{MAX}$;
    The initial set of physical neighbours nodes,
    $Neighbours$.
Export: getNode()
    Let getNode() return and remove a random node ID
    from the local node ID cache $Q_i$
Initialisation:
1: $Q_i \leftarrow Neighbours$
2: randomly trim $Q_i$ such that $|Q_i| \leq Q_{MAX}$
At each cycle:
3: $j \leftarrow$ getNode()
4: send a cache message to $j$, $\langle Q_i, \textbf{true} \rangle$
At event: received a cache message $\langle Q_j, r \rangle$ from $j$
5: if $r$ is true then
6:     send a cache message to $j$, $\langle Q_i, \textbf{false} \rangle$
7: end if
8: $Q_i \leftarrow Q_i \cup Q_j \cup \{ j \}$
9: randomly trim $Q_i$ such that $|Q_i| \leq Q_{MAX}$
```

Figure 2. The pseudocode of the Node Cache Protocol

The node selection protocol maintains a local cache Q of node identifiers (IDs), with $|Q| = Q_{MAX}$. The cache is initialised with the physical neighbours (lines 1-2). At each protocol cycle the local cache is sent to a node randomly chosen from the cache according to a uniform pdf (lines 3-4). When a remote cache is received, it is merged with the local one and trimmed to the maximum size by randomly removing a number of IDs exceeding Q_{MAX} (lines 5-9). The procedure can be considered a practical implementation of multiple random walks. After a sufficiently large number of cycles, the entries in the local cache are uniformly distributed. In regular connected graphs, random walks converge to uniform independent samples of the node set in a polynomial number of steps. In expander graphs, i.e., sparse graphs that are very well connected, random walks converge to the uniform distribution in $O(\log(n))$ [21].

The node cache protocol provides a local service *getNode()*, which removes and returns a random node from the cache.

IV. PERFORMANCE EVALUATION

In this section the proposed SPSP is evaluated and compared with PSP and PPG protocols for the decentralised approximate computation of a global average. At each cycle c of the aggregation protocol each node i computes an estimate $\tilde{m}$ of the global true average m:

$$\tilde{m}_i(c) \approx m = \frac{\sum_{j=1}^{n} v_j}{n} \qquad (2)$$

A. Experimental setting

We implemented the three protocols, SPSP, PSP and PPG, in an ad hoc simulator based on discrete events [22]. The simulator has an event scheduler, a set of processes, which simulate network nodes, a topology manager and events, which represent operations such as initialisation, messages, computation, etc. The simulations assume that a reliable point-to-point communication protocol is available in the network.

We have tested the distributed algorithms in two different types of network topologies:

- two Internet-like topologies were generated using BRITE [23] with a Waxman model to simulate a flat level Autonomous System with 1000 and 5000 nodes;
- two 2D mesh topologies were also generated with dimensions, respectively, 40×25 (1000 nodes) and 100×50 (5000 nodes).

The algorithms were evaluated according to the peak data distribution, where only a node i holds as local value $v_i = N$, and all others j hold as local value $v_j = 0$. As shown in Table I to compute the average each node i holds weight $w_i = 1$. According to this setting, m is equal to 1. We have tested all the discussed algorithms with two different peak data distribution randomly generated.

Each one of the tested protocols uses the Node Cache Protocol reported in Section III-A. In particular, each node has its own node cache with Q_{MAX} equal to 20.

In order to simulate the algorithms and to collect relevant performance indices, we have adopted an opportune cycle structure of fixed length where the aggregation is carried out. The cycle structure guarantees that there is no overlap in the communication of different cycles and provides a simple mechanism for varying the *atomic violation* percentage (AVP).

Each cycle is composed of four intervals as shown in Figure 3. The four intervals have a fixed length of, respectively, $d1$, $d2$, $d2$ and $d3$:

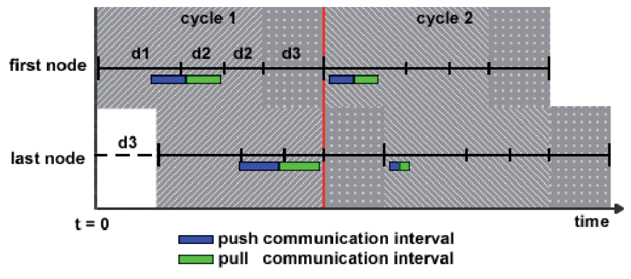

Figure 3. Cycle structure

- $d1$ is the length of the interval where nodes start push operations;
- $d2$ is the maximum propagation delay between any pair of nodes in the network;
- $d3$ is the maximum synchronisation offset between any pair of nodes in the network.

We assume a uniformly distributed synchronisation offset for the start of the aggregation process at different nodes. In all experiments the maximum synchronisation offset ($d3$) between any pair of nodes is set to 10 msec. This value is similar to a clock synchronisation offset, which can be obtained using e.g., NTP [24] or PariSync [25].

Each node (source) initiates a push operation at a random instant of the first interval ($d1$). In particular, $d1$ is the simulation parameter through which the AVP can be varied. By decreasing the value of $d1$ the AVP increases and vice versa.

After a propagation delay the push message is received at the destination node, which asynchronously replies with a symmetric push (pull) message. After the corresponding propagation delay the reply is received at the source node. The second and third intervals account for these propagation delays. Two intervals of length $d2$ are necessary to guarantee that a symmetric push operation (push-pull) is completed. The value of $d2$ is a property of the network topology.

Finally a padding interval ($d3$) is required to ensure that communications of two different cycles do not overlap because of the synchronisation offset of the nodes.

B. Analysis

The performance of the three methods (SPSP, PSP and PPG) are compared in terms of accuracy and convergence speed at different AVP levels. The accuracy is computed in terms of the mean percentage error of the estimated average among all nodes, as shown in equation 3.

$$\text{MPE}(c) = \frac{1}{n} \cdot \sum_{i=1}^{n} \left| \frac{m - \tilde{m}_i(c)}{m} \right| \qquad (3)$$

The convergence speed is evaluated by means of the variance of $\tilde{m}_i$ among all nodes over time, as indicated in equation 4.

$$\text{VAR}(c) = \frac{1}{n-1} \cdot \sum_{i=1}^{n} (m - \tilde{m}_i(c))^2 \qquad (4)$$

As shown in Figure 4, PPG is sensitive to AVP. PPG always reaches a non-null error for $AVP > 0\%$. With a very large $d1$ interval the smallest AVP level (0.3%) is obtained. Even in this case PPG does not converge to the true average (MPE $\neq 0$). PSP preserves the *mass conservation* invariant and is guaranteed to converge to the true average. However it has a slow convergence speed compared to the other protocols. SPSP is not sensitive to the AVP level and is guaranteed to converge to the true average. Moreover, SPSP provides the best accuracy in all simulated scenarios.

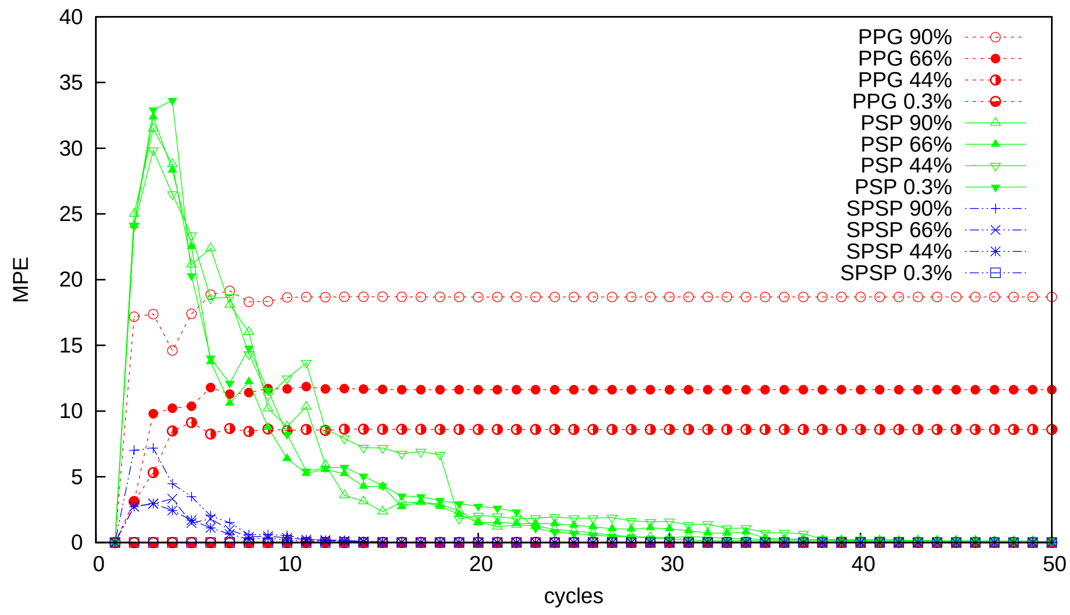

Figure 4. MPE varying the AVP w.r.t. 100 different simulations: BRITE and Mesh topologies, 1000 and 5000 nodes, 2 different Peak Data Distribution.

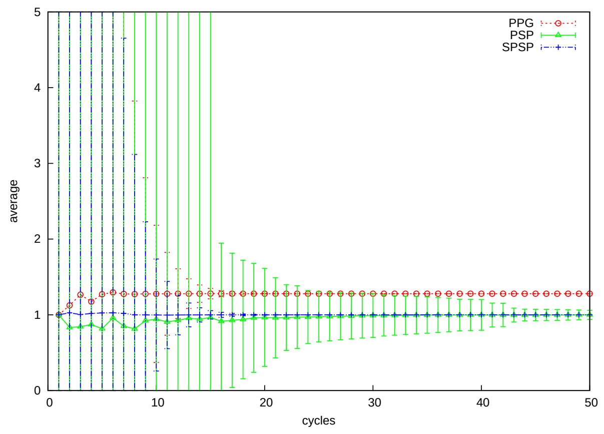

Figure 5. Average and standard deviation of the estimated aggregate over the network nodes. BRITE topology with 5000 nodes. AVP equal to 90%.

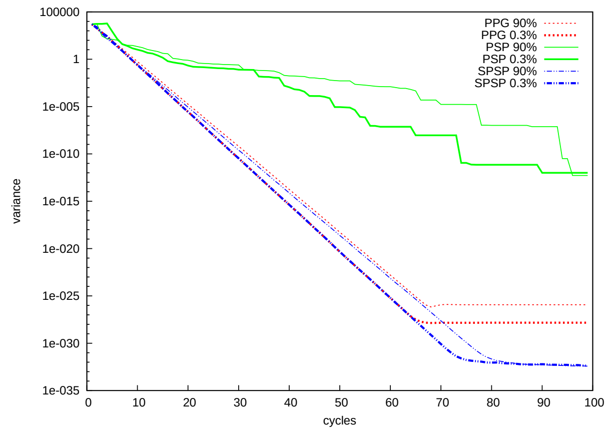

Figure 6. Convergence speed (variance). BRITE topology with 5000 nodes. AVP equal to 0.3% and 90%.

In Figures 5 and 6, PPG and SPSP have a similar variance trend, however PPG converges to an incorrect estimate $\tilde{m}$.

V. Conclusion

The Symmetric Push-Sum Protocol (SPSP) is a novel Gossip-based aggregation protocol that is suitable for computing aggregation functions on networks of any scale. The proposed algorithm is totally decentralized and robust and preserves the *mass conservation* invariant. The experimental analysis has confirmed that the algorithm outperforms the state of the art protocols (i.e., PSP and PPG) for the average aggregation function. In particular, SPSP does not violate the *mass conservation* invariant, similarly to PSP, and is much faster than PSP. SPSP and PPG have similar convergence speed; SPSP guarantees a convergence to the true global aggregate, while PPG does not. Future research directions will focus on the evaluation of the protocol in dynamic environments with churn rate and with node and link failures.

REFERENCES

[1] F. Cappello, S. Djilali, G. Fedak, T. Hérault, F. Magniette, V. Néri, and O. Lodygensky, "Computing on large-scale distributed systems: Xtremweb architecture, programming models, security, tests and convergence with grid," *Future Generation Comp. Syst.*, vol. 21, no. 3, pp. 417–437, 2005.

[2] M. Bawa, H. Garcia-Molina, A. Gionis, and R. Motwani, "Estimating aggregates on a peer-to-peer network," Stanford InfoLab, Technical Report 2003-24, April 2003.

[3] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tag: a tiny aggregation service for ad-hoc sensor networks," *SIGOPS Oper. Syst. Rev.*, vol. 36, pp. 131–146, December 2002.

[4] M. Dam and R. Stadler, "A generic protocol for network state aggregation," in *In Proc. Radiovetenskap och Kommunikation (RVK)*, 2005, pp. 14–16.

[5] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, "Epidemic algorithms for replicated database maintenance," in *Proc. of the sixth annual ACM Symposium on Principles of distributed computing*, ser. PODC '87. ACM, 1987, pp. 1–12.

[6] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking, "Randomized rumor spreading," in *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, 2000, pp. 565–574.

[7] B. Ghit, F. Pop, and V. Cristea, "Epidemic-style global load monitoring in large-scale overlay networks," *International Conference on P2P, Parallel, Grid, Cloud, and Internet Computing*, pp. 393–398, 2010.

[8] D. H. H. Sheng Di, Cho-Li Wang, "Gossip-based dynamic load balancing in an autonomous desktop grid," in *Proc. of the 10th International Conference on High-Performance Computing in Asia-Pacific Region*, 2009, pp. 85–92.

[9] Y. Ma and A. Jamalipour, "An epidemic P2P content search mechanism for intermittently connected mobile ad hoc networks," in *IEEE Global Telecommunications Conference (GLOBECOM)*, 2009, pp. 1–6.

[10] S. Tang, E. Jaho, I. Stavrakakis, I. Koukoutsidis, and P. Van Mieghem, "Modeling gossip-based content dissemination and search in distributed networking," *Comput. Commun.*, vol. 34, pp. 765–779, May 2011.

[11] N. Bansod, A. Malgi, B. K. Choi, and J. Mayo, "Muon: Epidemic based mutual anonymity in unstructured P2P networks," *Computer Networks*, vol. 52, no. 5, pp. 915–934, 2008.

[12] L. Chitnis, A. Dobra, and S. Ranka, "Aggregation methods for large-scale sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 4, pp. 1–36, April 2008.

[13] N. Marechal, J.-M. Gorce, and J. Pierrot, "Joint estimation and gossip averaging for sensor network applications," *IEEE Transactions on Automatic Control*, vol. 55, no. 5, pp. 1208–1213, may 2010.

[14] A. Dimakis, S. Kar, J. Moura, M. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1847–1864, nov. 2010.

[15] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, oct. 2003, pp. 482–491.

[16] M. Jelasity, A. Montresor, and O. Babaoglu, "Gossip-based aggregation in large dynamic networks," *ACM Transactions on Computer Systems*, vol. 23, pp. 219–252, August 2005.

[17] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2508–2530, june 2006.

[18] P. Jesus, C. Baquero, and P. Almeida, "Dependability in aggregation by averaging," in *1st Symposium on Informatics (INForum 2009)*, sept. 2009, pp. 482–491.

[19] W. Terpstra, C. Leng, and A. Buchmann, "Brief announcement: Practical summation via gossip," in *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing (PODC)*. ACM, August 2007, pp. 12–15.

[20] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Transactions on Automatic Control*, vol. 31, no. 9, pp. 803–812, sep 1986.

[21] D. Gillman, "A chernoff bound for random walks on expander graphs," *SIAM Journal on Computing (Society for Industrial and Applied Mathematics)*, vol. 27, no. 4, pp. 1203–1220, 1998.

[22] G. Fortino, C. Mastroianni, and W. Russo, "A hierarchical control protocol for group-oriented playbacks supported by content distribution networks," *Journal of Network and Computer Applications*, vol. 32, no. 1, pp. 135–157, 2009.

[23] A. Medina, I. Matta, and J. Byers, "On the origin of power laws in Internet topologies," *SIGCOMM Comput. Commun. Rev.*, vol. 30, pp. 18–28, April 2000.

[24] D. L. Mills, "On the accuracy and stability of clocks synchronized by the network time protocol in the Internet system," *ACM Computer Communication Review*, vol. 20, pp. 65–75, 1990.

[25] P. Bertasi, M. Bonazza, N. Moretti, and P. E., "PariSync: Clock Synchronization in P2P Networks," *ISPCS 2009 Internation IEEE Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, pp. 12–16, October 2009.