# Fault tolerant decentralised K-Means clustering for asynchronous large-scale networks

# Fault Tolerant Decentralised K-Means Clustering for Asynchronous Large-Scale Networks

Giuseppe Di Fatta[a], Francesco Blasa[b], Simone Cafiero[b], Giancarlo Fortino[b]

[a]*School of Systems Engineering, The University of Reading, Reading, UK*
*Email: g.difatta@reading.ac.uk*
[b]*Dipartimento di Informatica, Elettronica e Sistemistica, University of Calabria, Italy*
*Email: g.fortino@unical.it*

**Abstract**

The K-Means algorithm for cluster analysis is one of the most influential and popular data mining methods. Its straightforward parallel formulation is well suited for distributed memory systems with reliable interconnection networks, such as massively parallel processors and clusters of workstations. However, in large-scale geographically distributed systems the straightforward parallel algorithm can be rendered useless by a single communication failure or high latency in communication paths. The lack of scalable and fault tolerant global communication and synchronisation methods in large-scale systems has hindered the adoption of the K-Means algorithm for applications in large networked systems such as wireless sensor networks, peer-to-peer systems and mobile ad hoc networks. This work proposes a fully distributed K-Means algorithm (*Epidemic K-Means*) which does not require global communication and is intrinsically fault tolerant. The proposed distributed K-Means algorithm provides a clustering solution which can approximate the solution of an *ideal* centralised algorithm over the aggregated data as closely as desired. A comparative performance analysis is carried out against the state of the art sampling methods and shows that the proposed method overcomes the limitations of the sampling-based approaches for skewed clusters distributions. The experimental analysis confirms that the proposed algorithm is very accurate and fault tolerant under unreliable network conditions (message loss and node failures) and is suitable for asynchronous networks of very large and extreme scale.

*Keywords:* distributed clustering, k-means, peer-to-peer data mining, gossip protocols, epidemic protocols, extreme scale computing

## 1. Introduction

Emerging challenges in ubiquitous networks and computing [1] include the ability to extract useful information from a vast amount of data which are intrinsically distributed. However, the dynamic and unreliable nature of large-scale distributed environments pose hard challenges for parallel algorithms.

Research on Distributed Data Mining (DDM) has focused on the formulation of data mining algorithms for distributed computing environments, where each node processes its local data and contributes to compute a global solution. In many applications the solution is required to be available at every node. This is particularly important when considering applications in networked systems where each node is autonomous and active, like in peer-to-peer systems, mobile ad hoc networks, vehicular ad hoc networks, mobile social networks [2, 3], wireless sensor networks. Obviously it is desirable that the solutions at different nodes are identical or within a bounded approximation error.

For example, a number of projects have recently been devoted to the design and implementation of Decentralised Online Social Networks (DOSN) (e.g. *Diaspora, Tribler, Spar, Whats up, Scope, SuperNova, PrPl, OneSocialWeb*). DOSN are a typical scenario where fault-tolerant Distributed Data Mining applications may be required and successfully applied.

The goal of cluster analysis is the determination of groups (clusters) of data items which show high intra-cluster similarity and low inter-cluster similarity. Clustering is known to be an NP-hard problem [4, 5]. Therefore many clustering algorithms use heuristic approaches, such as iterative ‘hill climbing’ methods. These methods typically lead to local convergence, where further improvements cannot be made.

The K-Means algorithm [6, 7] is a typical ’hill climbing’ method for cluster analysis and, in general, is one among the most influential and popular data mining methods [8, 9]. K-Means determines a set of $K$ points, called centres or centroids, so as to minimise the mean-squared distance from each data point to its nearest centre. The K-Means algorithm is an iterative refinement process, where at each iteration the clustering assignments are updated, consequently changing the definition of the clusters and, consequently, of the centroids. Ideally during and after the execution of a distributed algorithm the meaning of each cluster should be consistent (synchronised) at every node of the systems.

Parallel and distributed algorithms which have been proposed to solve the distributed K-Means problem, have typically adopted global communication (global reduction and broadcast) at the end of each K-Means iteration to provide global synchronisation and consistency of the centroids at every node. However, in large-scale distributed systems global communication is obviously unrealistic. In fact, the lack of scalable and fault tolerant global communication and synchronisation methods has hindered the applications of data mining algorithms in large-scale distributed computing environments.

In this work we present a distributed stochastic formulation of the K-Means clustering algorithm. The proposed algorithm is fully decentralised and intrinsically fault tolerant. The global synchronisation and consistency required in the K-Means algorithm is achieved without global communication. The global reduction operation is performed by means of a Gossip-based aggregation protocol [10, 11, 12], which provides statistical guarantees of the convergence to the clustering result which would be obtained by a monolithic K-Means approach over the aggregated data. The same statistical guarantees ensure that the cluster centres are consistent within a bounded approximation error at every node of the system.

The simulations in the experimental analysis have confirmed the quality and consistency of the results. In particular the results show that the proposed approach always performs much better than the state of the art in distributed K-Means for large-scale systems [13, 14, 15, 16, 17] and its accuracy degrades gracefully under unreliable network conditions, such as packet loss and node failures.

The rest of the paper is organised as follows; Section 2 reviews related work on distributed K-Means Clustering. Section 3 recalls the terminology and theory of the K-Means algorithm in centralised and parallel settings. Section 4 introduces a stochastic formulation of the distributed K-Means algorithms for large-scale systems. Section 5 describes the Gossip-based aggregation protocol. Section 6 introduces the proposed decentralised and fault tolerant algorithm for the distributed K-Means problem. Section 7 presents experimental comparative analyses for both the adopted aggregation protocol and the distributed K-Means algorithm. Finally Section 8 provides some conclusive remarks and future research directions.

## 2. Related Work

The most notable effort for the formulation of distributed K-Means in large-scale systems has produced a few related variants [13, 14, 15, 16, 17] which are all based on two sampling strategies, local and random sampling.

The Local Synchronization-Based P2P K-Means (*Local P2P K-Means*) [14, 17] adopts a local strategy, where each node communicates and synchronises only with its physical neighbours.

The Random Sampling-based Peer-to-Peer K-Means (*Random P2P K-Means*) [15] adopts a random strategy, where each node communicates and synchronises with a randomly selected sample of network nodes. The local approach presented in [15] is similar to the one described in [14, 17] with a slightly different convergence criterion.

As these approaches did not provide an analytical accuracy guarantee, [17] introduces another approach, Uniform Sampling-based Peer-to-Peer K-Means (*USP2P K-Means*) which offers an accuracy guarantee. The guarantee holds if the network topology and data do not change during the execution of the algorithm. *USP2P K-Means* assumes that a leader node computes the clustering solution by synchronising and communicating with a uniform sample of nodes in a master-slave approach. The clustering solution is only computed at this single node and for this reason *USP2P K-Means* does not strictly solve the distributed K-Means problem and does not use a decentralised approach. When data are not uniformly distributed a large percentage of the nodes have to be involved in the communication in order to achieve a good approximation. Moreover, if the solution has to be made available to all nodes, the leader node has to broadcast the final result, or alternatively each node of the network needs to run an independent *USP2P K-Means* with two drawbacks: the clustering solution at different nodes may be different and the communication cost is greatly redundant.

Sampling approaches can provide a good approximation of the solution when the data clusters are uniformly distributed in the network. Under moderately or highly skewed cluster distributions these approaches are expected to fail in approximating the *ideal* solution and in guaranteeing consistency over the network nodes.

In [15] the performance of the algorithm is also tested under non-uniform data distributions. However, it should be noted that the tests are performed in a relatively small topology (50 nodes) and the skewed distribution refers to the number of data points at each node, not to the distribution of clusters over the nodes. Whether the data points of each cluster are randomly distributed is not specified.

In this work we explicitly address this issue and show that it has a significant impact on the performance of the sampling approaches. Their accuracy significantly degrades for a non-uniform cluster data distribution, which is a more realistic scenario for intrinsically distributed data. For example, in P2P networks it is expected to find geographic aggregations in the data distributions. They are based on deterministic communication patterns and do not naturally cope with unreliable and dynamic network conditions, such as packet loss and node failures.

The proposed algorithm is not based on a deterministic reduction operation over a sample of nodes and data. It rather adopts a stochastic aggregation operation based on an epidemic approach, which is intrinsically scalable and fault tolerant, and, above all, allows achieving an accuracy and a consistency of the results even for highly skewed cluster distributions.

## 3. The Deterministic K-Means Problem

### *3.1. Centralised K-Means*

Given a set $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ of $n$ data vectors in a $d$ dimensional space $\mathbb{R}^d$ and a parameter $K$ ($1 < K < n$) which defines the number of desired clusters, K-Means determines a set of $K$ vectors $\mathcal{M} = \{\boldsymbol{m}_1, \ldots, \boldsymbol{m}_K\}$, called centres or centroids, to minimise the average within-cluster variance. The clustering associated to the set of centroids $\mathcal{M}$ is the set of disjoint partitions $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_K\}$, such that $\bigcup_{k=1}^{K} \mathcal{P}_k = X$ and $\mathcal{P}_i \bigcap \mathcal{P}_j = \emptyset$ $(i \neq j)$.

The centroid of the cluster $k$ is derived to approximate the 'centre of mass' of the cluster partition $\mathcal{P}_k$ and is defined as:

$$\boldsymbol{m}_k = \left(\frac{1}{n_k}\right) \sum_{i=1}^{n_k} \boldsymbol{x}_i^{(k)}, \quad (1)$$

where $n_k$ is the number of data points in the cluster $k$ ($n_k = |\mathcal{P}_k|$), and $\boldsymbol{x}_i^{(k)}$ is a data point in the cluster $k$ ($\boldsymbol{x}_i^{(k)} \in \mathcal{P}_k$).

The error for each cluster is the squared sum of a norm $\|\cdot\|$, e.g. the Euclidean norm, between each input data point and its nearest centroid. The objective function that K-Means optimises is the overall error (square-error distortion) $E(M)$ which is given by the sum of the squared errors for each cluster:

$$E(M) = \sum_{i=1}^{n} \min_{k=1..K} \|\boldsymbol{x}_i - \boldsymbol{m}_k\|^2 = \sum_{k=1}^{K} \sum_{i=1}^{n_k} \left\|\boldsymbol{x}_i^{(k)} - \boldsymbol{m}_k\right\|^2 . \quad (2)$$

The sum of the squared errors is a popular objective function as it combines a measure of homogeneity and separation of the clusters. The optimal clustering corresponds to the minimum sum of the squared errors. For general values of $K$ and $d$ and with the Euclidean distance as metric, the problem is known to be NP-hard [5].

Given an initial condition, i.e. the initial set of centroids, the K-Means algorithm [6, 7] adopts a 'hill climbing' heuristic method to determine the local minimum of the objective function. The algorithm repeats an iterative refinement step until no further improvement can be achieved. At each iteration two main steps are performed:

- **distance calculation**: for each data point compute the distance to each cluster centroid and find the closest cluster centroid;

- **centroid update**: recompute each cluster centroid as the average of data points assigned to the cluster partition.

### *3.2. Parallel K-Means*

Parallel clustering algorithms have been extensively studied (e.g. [18], [19], [20]). In particular, [20] proposes a straightforward implementation of the brute force K-Means algorithm for distributed memory systems, which is based on a master-slave approach and static data partitioning. The input data points are partitioned in equal sized sets and distributed to the processes. Initial centroids are generated at the master and distributed (broadcast) to the other processes. Each process performs a K-Means iteration on its local data partition. At the end of each iteration a global reduction operation (e.g. *MPI_ALLREDUCE* [21]) generates the updated global centroid vectors and the global distortion measure:

$$\boldsymbol{m}_k = \frac{\sum_{i=0}^{P-1} \boldsymbol{s}_{i,k}}{\sum_{i=0}^{P-1} n_{i,k}}, \qquad 1 \le k \le K \quad (3)$$

$$E(M) = \sum_{i=0}^{P-1} e_i \quad (4)$$

where $P$ is the number of parallel processes, $s_{i,k}$ and $n_{i,k}$ are, respectively, the sum and the number of data points in the local partition at process $i$ which have been associated with cluster $k$; $e_i$ is the local contribution at process $i$ to the global square-error distortion.

At each process and for each iteration three main steps are performed:

- **distance calculation**: for each data point of the local data set compute the distance to each cluster centroid and find the closest cluster centroid;

- **global reduction operation**: all nodes perform a deterministic reduction operation to compute the global sums and counts for each cluster partition and the global error;

- **centroid update**: recompute each cluster centroid as the average of all data points assigned to the cluster using the global sums and counts.

At each iteration a single all-reduce operation can be used to compute the aggregation for $(d \cdot K + 1)$ real values and $K$ integer values over all processes. The number of communication steps of the all-to-all reduction operation is $log_2(P)$.

In a static and homogeneous computing environment the approach in [20] guarantees a perfectly balanced load among the processes. In [22, 23] parallel formulations of the efficient K-Means algorithms based on KD-trees were investigated. All these parallel approaches are based on a global deterministic reduction operation. They are not suitable for large-scale geographically distributed systems.

## 4. Stochastic K-Means for Large-Scale Distributed Systems

Given a set of $P$ nodes connected by a physical network with a transmission protocol for point-to-point communication, the logic communication network can be represented by a completely connected graph, where the edge between two nodes is associated with the communication cost of the shortest path in the physical network.

Each node $i$ of the network has a local set of data $X_i = \{\mathbf{x}\}$ ($0 \le i < P$), where $\mathbf{x} \in \mathbb{R}^d$. The equivalent centralised clustering problem is defined over the aggregated data set $X = \bigcup_{i=0}^{P-1} X_i$.

Given a set of $K$ initial centroids, the sequential K-Means deterministically converges to a set of final centroids $\mathcal{M}^{(*)} = \{\mathbf{m}_k^{(*)}\}$ which locally minimise the objective function (2). These centroids uniquely define the resulting partitional clustering configuration $\mathcal{P}^{(*)} = \{\mathcal{P}_1^{(*)}, \ldots, \mathcal{P}_K^{(*)}\}$. We refer to $\mathcal{M}^{(*)}$ and $\mathcal{P}^{(*)}$, respectively, as the *ideal* centroids and the *ideal* cluster partitions over the aggregated data.

A stochastic distributed K-Means problem can be defined by relaxing the need of explicitly computing and synchronising the unique global centroids for all network nodes. Global centroids are still implicitly defined, but no longer computed in a deterministic way.

Given a unique set of $K$ initial centroids, each node $i$ determines a set of $K$ local centroids $\mathcal{M}^{(i)} = \{\mathbf{m}_k^{(i)}\}$ which approximate the *ideal* ones with a probabilistic guarantee. The set of local centroids uniquely defines a set of disjoint partitions $\mathcal{P}^{(i)} = \{\mathcal{P}_k^{(i)}\}$ $(1 \leq k \leq K)$ of the local data, where $\bigcup_{k=1}^{K} \mathcal{P}_k^{(i)} = X_i$.

If the stochastic approximation of the centroids at various network nodes is independent and small, then the aggregated local data partitions closely approximate the *ideal* cluster partitions over the aggregated data and the distributed solution closely approximates the minimisation of the global optimisation function (2).

The stochastic nature of the local centroids introduces local approximation errors. However, globally the aggregated cluster partitions maintain the properties and structure of the *ideal* solution.

In the stochastic definition of the K-Means problem, we are interested in characterising the error between the local centroids and the *ideal* ones and the variance of the local centroids at different nodes.

The error with respect to the equivalent centralised problem is defined as the error of the average distributed K-Means solution with respect to the *ideal* solution. At node $i$ the error is:

$$err_{cKM}(i,k) = |\mathbf{m}_k^{(i)} - \mathbf{m}_k^{(*)}|.$$

The mean squared error of the distributed K-Means solutions w.r.t. the *ideal* solution is given by

$$MSE_{cKM} = \frac{1}{P \cdot K} \sum_{i=0}^{P-1} \sum_{k=1}^{K} |\mathbf{m}_k^{(i)} - \mathbf{m}_k^{*}|^2. \tag{5}$$

The variance of the above error corresponds to the difference between the solutions obtained at different network nodes. For nodes $i$ and $j$ $(i \neq j)$ the consistency error is defined as:

$$err_{dKM}(i,j,k) = |\mathbf{m}_k^{(i)} - \mathbf{m}_k^{(j)}|.$$

The mean square consistency error at node $i$ is given by

$$mse_{dKM}(i) = \frac{1}{(P-1) \cdot K} \sum_{j=0, j\neq i}^{P-1} \sum_{k=1}^{K} |\mathbf{m}_k^{(i)} - \mathbf{m}_k^{(j)}|^2.$$

The overall consistency error in the distributed system is given by:

$$MSE_{dKM} = \frac{2}{P \cdot (P-1) \cdot K} \sum_{i=0}^{P-2} \sum_{j=i+1}^{P-1} \sum_{k=1}^{K} |\mathbf{m}_k^{(i)} - \mathbf{m}_k^{(j)}|^2. \tag{6}$$

Another important performance metric is the percentage of cluster membership mismatch (PMM) w.r.t. the *ideal* solution at each node, which is defined as:

$$PMM(i) = \frac{\sum_{k=1}^{K} |\mathcal{P}_k^{(i)} \bigcap \mathcal{P}_k^{(*)}|}{|\bigcup_{k=1}^{K} \mathcal{P}_k^{(i)}|}. \tag{7}$$

The average clustering accuracy over all network nodes is defined[1] as:

$$Accuracy_{dKM} = \frac{\sum_{i=0}^{P-1} PMM(i)}{P}. \tag{8}$$

## 5. Gossip-based Aggregation Protocol

Epidemic or Gossip-based protocols are a robust and scalable communication paradigm to disseminate information (broadcasting) in a large-scale distributed environment using randomised communication [24, 25]. The advantages of Gossip protocols over global communication schemes based on deterministic interconnection networks are their inherent robustness and scalability.

Gossip protocols can also be adopted to solve the data aggregation problem in a fully decentralised manner. Each node $i$ of a networked system holds some local value $x_i$ and needs to compute a global aggregation function $F(x_0, \cdots, x_{P-1})$ (e.g. minimum, maximum, count, sum, average, etc.).

At each cycle of a Gossip protocol each node independently selects a random communication peer with a uniform probability distribution. Local states are exchanged and updated. Typically, the update of the local state produces a reduction in the variance of the aggregate estimate. The definition of local state, the number and type of messages and the update operation (variance reduction operation) depend on the particular aggregation protocol.

An approximation of the global aggregate function can be obtained at every node within a fixed number of protocol cycles. The correct convergence to the true global value, for example for computing the global average, is guaranteed if the 'mass conservation' invariant is maintained, i.e. at any time the sum of all aggregated values is constant [10].

The diffusion speed of a Gossip-based aggregation protocol 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.

In the implementation of the proposed distributed K-Means algorithm we have adopted a novel and straightforward Gossip-based aggregation protocol, which combines the advantages of the Push-Sum Protocol (PSP) and of the Push-Pull Gossip (PPG) protocol.

### 5.1. Push-Sum Protocol

The Push-Sum Protocol (PSP) [10] adopts an asynchronous and asymmetric approach. At each protocol cycle every peer pushes its local state to a random peer. A peer receiving a *push* message performs a variance reduction operation on its local state.

In uniform Gossip the probability of a peer being selected as destination of a *push* operation follows a binomial distribution. The probability of a peer not being selected at all during a protocol cycle is not zero. In this case the peer does not perform the variance reduction step during the cycle. For example, for a network with 1000 nodes, the probability of a peer receiving 0, 1 or 2 *push* messages during a cycle is, respectively, 36.7%, 36.8% and 18.4%.

[1]The definitions of PMM and average clustering accuracy are equivalent to the ones given in [17].

The diffusion speed of the simplest aggregation protocol has been shown to have a complexity $O(log(P)+log(\varepsilon^{-1})+log(\delta^{-1}))$ [10], where $P$ is the network size, $\varepsilon$ ($> 0$) is the maximum approximation error and $\delta$ ($> 0$) is the maximum probability of greater error than $\varepsilon$.

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 [10]. (For example, the global sum can be computed by setting the initial weight to 1 at a single node and to 0 in all others.) The global aggregate value is given by $\frac{v}{w}$ after a fixed number of communication cycles. At each cycle each node halves its local value and weight ($< v, w >=< \frac{v}{2}, \frac{w}{2} >$) and sends them to a random node. The global mass is guaranteed to be conserved in case a loss-less transport protocol is used. The number of messages which are sent in total at each cycle is $P$.

### *5.2. Push-Pull Gossip protocol*

The Push-Pull Gossip (PPG) protocol [11, 12] adopts a symmetric approach and each *push* from a source to a destination is followed by a reply message (*pull*). Both peers perform a variance reduction step. In PPG, $2 \cdot P$ messages are sent in total at each cycle. The probability that a peer does not perform a variance reduction step during a cycle is zero. As a consequence, the convergence rate is faster and steadier w.r.t. PSP. It is faster because the protocols uses twice the number of message at each cycle and steadier because every single peer is involved in at least one variance reduction operation. In general, push-and-pull schemes are expected to convergence faster than push-only schemes even with the same number of exchanged messages [25].

In PPG at each cycle a node $i$ chooses a random node $j$ to exchange their local current aggregation values. Each peer then performs an averaging operation $\frac{x_i+x_j}{2}$ to update its local value. In the description of the protocol ([11, 12]) these three steps are not required to be atomic and, in general, could be implemented with asynchronous communication. Apparently, in order to guarantee the mass conservation invariant, the only requirement would be that the two messages (push and pull) are both successfully received within the current Gossip cycle. However, in [26, 27] it has been pointed out that this is not the case. In order to conserve the global mass and to guarantee a convergence to the true aggregate value, the two messages used to exchange the local values must be sent and received in an atomic communication operation. This requirement imposes additional complexity in practical implementations and introduces dependability issues [26, 27] in real asynchronous environments.

### *5.3. Symmetric Push Sum Protocol*

We have adopted a Symmetric PSP protocol (*SPSP*) where $2 \cdot P$ messages are sent in total at each cycle. At each random push an asynchronous reply follows with a symmetric push. Differently from the pull in the push-and-pull scheme, here the push reply is not required to be performed atomically with the initiating push operation. In our preliminary analysis this variant, similarly to PPG, has shown better performances than PSP and is simpler to implement in a real asynchronous distributed system than PPG.

The theoretical analysis of the convergence of PPG [11, 12] is derived and is valid only for the ideal case with conservation of the global mass and for static and faultless networks. Under these conditions *SPSP* is equivalent to PPG and the results of the theoretical analysis still hold.

The description of the protocol is given in terms of communication cycles. The use of synchronous cycles simplifies the description and the analysis. The protocols has been implemented and tested in a completely asynchronous simulation environment.

In real environments independent local Poisson clocks can be used to generate synchronous cycles in asynchronous distributed environments (e.g. [28]). A second interesting approach different from the use of synchronous cycles, is the use of an exact global estimation of the right termination time, similar to the median-counter algorithm [25] for rumour spreading.

In the proposed *Epidemic K-Means* algorithm the aggregation protocol is used to compute several multi-dimensional weighted averages and a sum as discussed in the Section 6. The global sum is often used as example in previous work and, for the sake of brevity, here we provide only the formulation of the weighted average.

Assuming that each node $i$ holds a local value $x_i$ and a local weight $\omega_i$ ($\omega_i \geq 0$), the aggregation protocol performs a decentralised approximate computation of the global weighted average:

$$\tilde{m} \approx m = \frac{\sum_{i=0}^{P-1} \omega_i \cdot x_i}{\sum_{i=0}^{P-1} \omega_i}$$

The aggregation protocol is described in Figure 1. At each cycle a node $i$ halves its local value and weight ($< v, w >=< \frac{v}{2}, \frac{w}{2} >$) and sends them to a random node $j$. 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 $i$. Then, it adds the received value and weight to its own. In case node $i$ receives the symmetric push from $j$ immediately after its push operation (i.e. there is no interleaving message from other nodes), this mechanism is equivalent to PPG. Nevertheless, the symmetric push mechanism does not require atomicity of the two messages involved in the exchange to guarantee the mass invariant.

A good approximation of the global weighted average is available from a local service *getAggregate*() after a given number of cycles (*NC*).

The service *getAggregate*() is used to implement other services. The aggregation of the weighted average can be easily extended to multi-dimensional data. The service *getGlobalwAvg*($\boldsymbol{m}$, $w$) returns a weighted average of the distribution of local vectors $\boldsymbol{m}$. The service *getGlobalSum*(x) returns the global sum of local scalar values $x$. The sum operation can also be implemented with *getAggregate*() and requires that all initial weights are set to 0, apart from one which has to be set to 1. This initialisation operation is equivalent to a leader election operation, which could also be solved dynamically by means of

```
At node i
Require:
The initial local value $v_0$ and weight $w_0$
The number of cycles nc
Initialisation:
1   Set $< v, w > = < v_0, w_0 >$.
At each cycle:
1   $j \leftarrow getNode()$;
2   send an aggregation message to $j$:
    aggrMsg($< \frac{v}{2}, \frac{w}{2} >$, reply=true);
3   $v = \frac{v}{2}$ and $w = \frac{w}{2}$;
At event: received an aggregation message m from j
1   if m.reply is true
    1.1 send aggrMsg($< \frac{v}{2}, \frac{w}{2} >$, reply=false) to j;
    1.2 $v = \frac{v}{2}$ and $w = \frac{w}{2}$;
2   $v = v + m.v$;
3   $w = w + m.w$;
Export:
A local service getAggregate($v_0, w_0, nc$) which returns
the global aggregate $\frac{v}{w}$ after nc cycles.
```

Figure 1: The pseudocode of the aggregation protocol

a Gossip protocol. However, for the sake of simplicity, we have statically assigned weight 1 to a random node.

### *5.4. Node Cache Protocol*

In uniform Gossip protocols the random node selection is a critical operation. A global knowledge of the whole network at each node is not a reasonable assumption. We only assume that the network topology is a connected graph, each node knows its physical neighbours ($Neighbours()$) and a routing protocol is available.

Scalable membership management approaches have typically been adopted in Gossip-based protocols. The approach described in this section has been chosen for its simplicity and low communication cost. In a preliminary analysis, not reported in this work, no significant improvement was found in adopting more accurate and complex approaches (e.g. [29]). Nevertheless, the proposed *SPSP* aggregation protocol and, consequently, the *Epidemic K-Means* algorithm do not depend on a particular peer membership management protocol.

The node selection protocol maintains a local cache $Q$ of node identifiers (IDs), with $|Q| \leq QMAX$[2]. The cache is initialised with the physical neighbours. At each protocol cycle the local cache is sent to a node chosen from the cache at random with uniform probability. When a remote cache is received, it is merged with the local one and trimmed (at random) to the maximum size.

The procedure in Figure 2 is a practical implementation of multiple random walks. After sufficiently many 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 $O(log(P))$ [30].

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

```
At node i
Assumptions:
Let $Neighbours()$ return the set of physical neighbour nodes.
Let $Q_i$ be the local node ID cache, $|Q_i| \leq QMAX$.
Let getNode() return and remove a random node ID from $Q_i$.
Initialisation:
1   $Q_i \leftarrow Neighbours()$;
2   Randomly trim $Q_i$ such that $|Q_i| \leq QMAX$;
At each cycle:
1   $j \leftarrow getNode()$;
2   send cacheMsg($Q = Q_i$, $reply = true$) to $j$;
At event: received cache message m from node j
1   if m.reply is true,
    send cacheMsg($Q_i$, $reply = false$) to $j$;
2   $Q_i \leftarrow m.Q \bigcup \{j\} \bigcup Q_i$;
3   Randomly trim $Q_i$ such that $|Q_i| \leq QMAX$;
```

Figure 2: The pseudocode of the node cache protocol

## 6. The Epidemic K-Means Algorithm

This section presents a new method, the *Epidemic K-Means* algorithm, to solve the stochastic distributed K-Means problem. The proposed algorithm is ideal for very large scale distributed systems as it is highly scalable and fault tolerant.

The method does not employ global communication as it does not rely on any global communication operation (e.g. broadcast, all-to-all reduction). The algorithm adopts a local communication pattern, in the sense that during each K-Means iteration the number of communication operations initiated by each node is a small constant.

In the following we assume that a global set of initial centroids is available to all network nodes.

There are at least two different methods that can be used to achieve a global synchronisation of the initial centroids. In the first and simpler method each node uses the random number generator which is initialised with the same seed. Without loss of generality we can assume that data attributes are normalised in $[0, 1]$ and that the same initial centroids can be chosen uniformly at random in the domain $[0, 1]^d$ by all nodes.

This approach may not give sufficient flexibility and control, when for example different initial centroids need to be chosen for multiple runs of the K-Means algorithm over the same data. Even the solution of having multiple seeds configured a priori at the nodes may not be flexible enough. For example, the structure of the data in the multi-dimensional space may be such that

[2]In the experimental analysis $QMAX$ has been set to 20.

initial centroids chosen uniformly at random over the entire domain is a poor start for K-Means. A safer and more popular initialisation method for the centroids is a random choice over the input data points. In this case a Gossip-based protocol can be used to implement a global random sampling method [10] to initialise the centroids. In our tests we adopted the simpler first method.

The *Epidemic K-Means* algorithm substitutes the deterministic all-reduce operation of the parallel K-Means algorithm with a stochastic aggregation phase. Under certain conditions (mass conservation and static data and network) the Gossip protocol provides statistical guarantees that the aggregation process converges to the true global aggregate (within a small bounded error) at every node of the systems. The efficiency and correctness of this phase is at the core of the *Epidemic K-Means* algorithm. The algorithm performance depends on how efficient, effective, scalable and resilient the Gossip protocol is.

The *Epidemic K-Means* algorithm is described in Figure 3. The algorithm requires a local set of data $X_i$, a set of initial centroids $M$ and a parameter $\tau$ which defines the termination condition in terms of the relative improvement of the global error at each iteration.

At any time each node may have one of two states *ACTIVE* and *CONVERGED*. When $ACTIVE$ a node performs two phases at each K-Means iteration. In the first phase (lines $5-13$) computation over the local data set is performed similarly to the sequential K-Means. For each data point $x$ the closest centroid is computed and the local cluster partitions are determined. For each cluster the local partial sums and counts are computed (lines $9-12$). The local squared error is also computed (line 13).

In the second phase global aggregates are computed by means of the Gossip-based aggregation protocol (lines $14-17$). For each cluster the global weighted average is determined by means of the local service *getGlobalwAvg*() and the centroid is updated accordingly (line 15). The global squared error is computed by means of the local service *getGlobalSum*() (line 17).

The K-Means iteration is repeated until the global error is improved of at least a specified minimum threshold $\tau$ (line 18). Below this threshold the node state is changed into *CONVERGED* and the node may still receive messages to which it will reply with a *reject* message. A node receiving a *reject* message is forced to the *CONVERGED* state.

We have chosen the threshold $\tau$ to be much greater than the maximum approximation error of the global aggregation of the objective function (2). In all our tests all nodes converged spontaneously at the same iteration and there is no need to explicitly synchronise the termination of the K-Means algorithm over the network. However, due to the stochastic nature of the Gossip-based aggregation we still need to consider the possibility that some node (with low probability) fails to approximate correctly the objective function.

A more robust termination condition has to be devised for the use of the approach in real environments with adversarial failures.

The algorithm requires several global aggregates: $K$ multi-dimensional weighted averages and a sum. In the implementation these have been combined into a single aggregation operation to optimise the communication cost. The data structure exchanged during the Gossip protocol cycles is a tuple of $K\cdot(d+1)+2$ elements. $K\cdot d$ elements are for the $d$ components of each vector $\boldsymbol{s}_{i,k}$, $K$ elements for the counts $c_{i,k}$ ($c_{i,k}$ is the associated weight for all $d$ components of $\boldsymbol{s}_{i,k}$), 2 elements for the local error and its associated weight.

```
Assume: Local node $i$
Require: $X_i, \mathcal{M}, \tau$
    The local data set $X_i = \{\boldsymbol{x}\}$, $n_i = |X_i|$
    The initial centroids $\mathcal{M}$, $|\mathcal{M}| = K$
    The termination threshold (%) $\tau$
 1: $state \Leftarrow ACTIVE$
 2: $\tilde{E} \Leftarrow \infty$
 3: repeat
 4:     $E_{prev} \Leftarrow \tilde{E}$
 5:     for all $\boldsymbol{x} \in X_i$ do
 6:         find closest centroid $\boldsymbol{m}_k \in \mathcal{M}$
 7:         assign $\boldsymbol{x}$ to cluster $C^k$
 8:     end for
 9:     for all $k$ such that $0 \le k < K$ do
10:         compute local sums $\boldsymbol{s}_{i,k} = \frac{\sum_{x\in C^k} \boldsymbol{x}}{|C^k|}$
11:         compute local count $c_{i,k} = |C^k|$
12:     end for
13:     compute the local sum of squared errors:
        $e_i = \sum_{k=0}^{K-1} \sum_{x\in C^k} |d(\boldsymbol{x}, \boldsymbol{m}_k)|^2$
14:     for all $k$ such that $0 \le k < K$ do
15:         compute the global weighted average and update
            the local centroid: $\boldsymbol{m}_k = getGlobalwAvg(\boldsymbol{s}_{i,k}, c_{i,k})$
16:     end for
17:     compute the global error:
        $\tilde{E} = getGlobalSum(e_i)$
18: until $\frac{E_{prev}-\tilde{E}}{E_{prev}} \ge \tau$
19: $state \Leftarrow CONVERGED$
```

Figure 3: The *Epidemic K-Means* algorithm

## 7. Experimental Analysis

We carried out the experimental tests in an ad hoc network simulator [31] based on discrete events. The simulator has an event scheduler, a set of processes which simulate network nodes, a topology manager and events which represent a number of operations, such as initialisation, messages, computation, etc.

The simulations in sections 7.1 and 7.2 assume that the network is static and that a loss-less point-to-point communication protocol is available. The simulations in sections 7.3 consider network failures, such as message losses and node failures. Each physical network connection is associated with a cost which is the maximum transmission delay of a message. For each message traversing a link a stochastic delay is determined between a minimum and a maximum percentage of the link cost ($MIN = 10\%$ and $MAX = 100\%$). The overall transmission delay of a message is determined as the sum of the link transmission delays over the shortest path between source and destination.

We have tested the algorithms in two different types of network topologies. Two Internet-like topologies were generated using BRITE [32] with a Waxman model to simulate a flat level Autonomous System[3] with 1000 and 2000 nodes. Two 2D mesh topologies were also generated with dimensions, respectively, $40 \times 25$ (1000 nodes) and $50 \times 40$ (2000 nodes).

No significative difference in performance has been noticed between networks of different size or type. Thus, the results presented in the following sections report global averages over multiple simulations (with different initialisation) over each of the different network topologies.

In the next Section an analysis of accuracy and the convergence speed of the adopted aggregation protocol is carried out. In the following Sections a comparative analysis of the distributed K-Means approaches is provided and, finally, an analysis of the performance degradation under unreliable network conditions is presented.

## 7.1. Gossip-based Aggregation

In this section the adopted Gossip-based protocol *SPSP* is evaluated and compared with *PSP* and *PPG* for the decentralised approximate computation of a global average in an asynchronous distributed environment. All three protocols have been implemented with non-blocking communication operations. All protocols adopt the Node Cache Protocol reported in Section 5.4, with $QMAX = 20$.

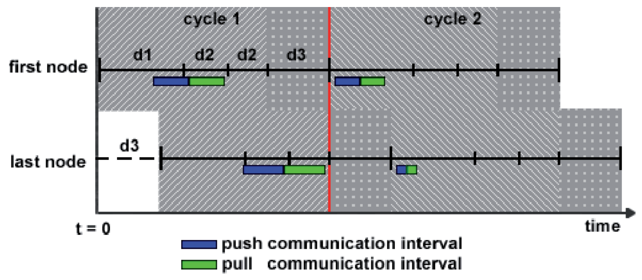


Figure 4: Gossip cycle structure

At each cycle $c$ of the aggregation protocol each node $i$ computes an estimate $\tilde{m}$ of the global true average $m$. The protocols were evaluated with a peak data distribution, where only a node $i$ holds a local value $v_i = N$, and all others $j$ hold a local values $v_j = 0$. According to this setting, the target value $m$ is equal to 1.

In order to simulate an asynchronous environment and collect relevant performance indices, we have adopted a cycle interval of variable length. A minimum cycle duration guarantees that there is no overlap in the messages of subsequent cycles and provides a simple mechanism for varying the percentage of *atomic violations* (AVP) in the variance reduction operations.

[3]BRITE was configured with the same parameters as used in [17].

Each cycle is composed of four time intervals as shown in Figure 4. The four intervals have a length of, respectively, $d1$, $d2$, $d2$ and $d3$:

- $d1$ is the (variable) length ($[0, 600]$s) of the interval in which nodes send push messages;
- $d2$ is the maximum propagation delay ($[0.5, 1]$s) between any pair of nodes in the network and is fixed for each network topology;
- $d3$ is a fixed maximum synchronisation offset ($10ms$) 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 $10ms$. This value is similar to a clock synchronisation offset, which can be obtained using e.g. NTP [33] or PariSync [34].

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. The interval $d1$ was varied between $0s$ and $600s$ to obtain the widest range for the AVP rates from almost 100% to almost 0%.

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 $d2$ was set according to each network topology and varied between $500ms$ and $1s$.

A fixed padding interval $d3$ ($10ms$) is required to ensure that communications of two different cycles do not overlap because of the synchronisation offset of the nodes.

### 7.1.1. Performance Analysis

The performance of the three protocols 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 over all nodes, as shown in equation 9.

$$\text{MPE}(c) = \frac{1}{n} \cdot \sum_{i=1}^{n} \left| \frac{m - \tilde{m}_i(c)}{m} \right| \quad (9)$$

The convergence speed is evaluated by means of the variance of $\tilde{m}_i$ over all nodes, as indicated in equation 10.

$$\text{VAR}(c) = \frac{1}{n-1} \cdot \sum_{i=1}^{n} (\mu_{\tilde{m}} - \tilde{m}_i(c))^2 \quad (10)$$

The accuracy w.r.t the true aggregate of the protocols can be compared in Figure 5. The accuracy of PPG is effected by the AVP level; PPG does not converge to the true aggregate for $AVP > 0\%$. The smallest AVP level (0.3%) is obtained with a

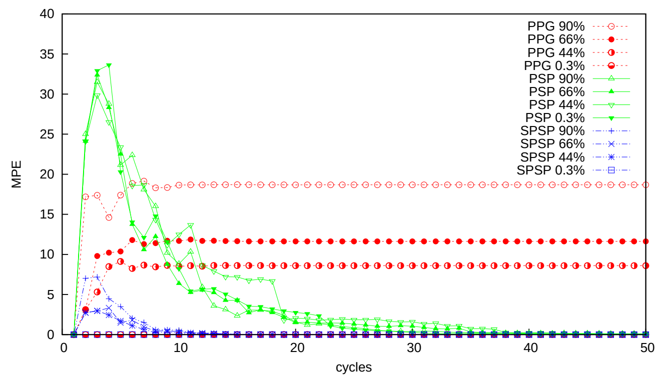


Figure 5: Mean percentage error (MPE) varying AVP level: averages over 100 different simulations.

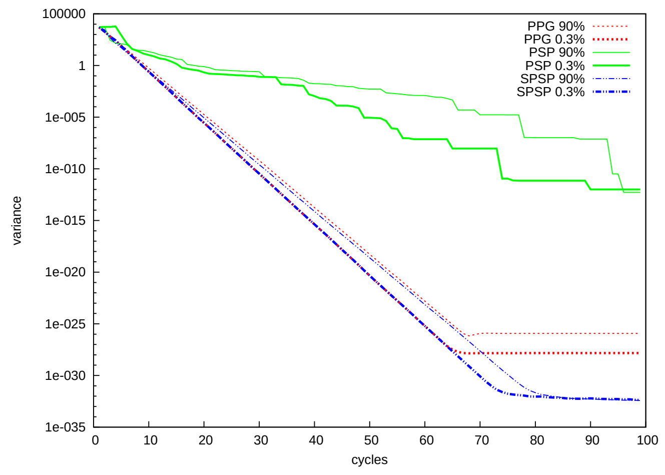


Figure 6: Convergence speed (variance). Selected *AVP* levels: 0.3% and 90%.

very large $d1$ interval (600$s$). Even in this case PPG does not converge to the true average (MPE $\neq 0$). Communications are performed randomly within the interval $d1$. A small number of violations of the mass conservation invariant are inevitable and a small error is still present after 100 cycles (though not visible in the chart).

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 effected by the AVP level and is guaranteed to converge to the true average after a sufficient number of cycles ($NC > 10$ in this case).

Figure 6 shows the convergence speed of the protocols. PPG and *SPSP* have a similar variance reduction slope; their convergence speed is slightly effected by the AVP rate. Their exponential convergence is faster than PSP, though PPG converges to an incorrect estimate.

In the following sections the simulations of the *Epidemic K-Means* are based on *SPSP* with the smallest Gossip interval ($d1 = 0s$) and a number of cycles $NC = 15$.

## 7.2. Distributed K-Means Clustering

In this section the proposed *Epidemic K-Means* algorithm is compared with two approaches based on sampling strategies, *Local P2P K-Means* [14, 17] and *Random P2P K-Means* [15].

For a given set of initial centroids the sequential K-Means clustering algorithm is fully deterministic. We have used the *ideal* clustering solution found by the sequential algorithm over the aggregated data till full convergence is reached, to determine and compare the clustering accuracy of the distributed algorithms.

### 7.2.1. Data

In this work an important distinction between data distribution in the multi-dimensional space and cluster distribution over the network is made.

A prototype for each cluster ($K = 20$ and $d = 10$) is generated with uniform probability distribution in the range $[0, 1]^d$. Data points are generated in $[0, 1]^d$ using independent multivariate Gaussian distributions ($\sigma = 0.2$) centered at each prototype. The total number of data points was varied to have always 100 data points per node.

The data were partitioned and assigned to network nodes in order to generate a range of cluster distributions, from uniformly distributed to highly skewed. In particular, the following procedure was used.

Each cluster is associated with a specific node, the cluster topological centre. $K$ cluster topological centres $n^{(k)}$ are selected randomly with uniform distribution among the network nodes. Data points of a cluster $k$ are distributed over network nodes with a probability which decreases with the distance from the centre $n^{(k)}$.

The likelihood and the probability that a node $n$ has a data point in the cluster $k$ whose topological centre is $n^{(k)}$ are defined as:

$$\begin{aligned} likelihood(n,k) &= (1 + distance(n, n^{(k)}))^{-\lambda} \\ Prob(n,k) &= \frac{likelihood(n,k)}{\sum_{j=0}^{K} likelihood(n,j)} \end{aligned} \tag{11}$$

where $\lambda \geq 0$ is a parameter which determines the cluster distribution. For $\lambda = 0$ the clusters are uniformly distributed in the network nodes. For larger values of $\lambda$ the clusters become less uniform and more concentrated in the neighbourhood of the topological cluster centres.

While the parameter $\lambda$ has been used to generate multiple distributed data sets with different cluster distributions, the Jain fairness index [35] has been used to measure the degree of non-uniformity of the cluster distribution. The Jain index is a more appropriate index and has good properties, which do not depend on the procedure used to generate and distribute the data.

The Jain index is often adopted to measure the equality of the allocation of a shared resource to different contending entities. The Jain index is defined as:

$$J = f(x_0, ..., x_{P-1}) = \frac{\left(\sum_{i=0}^{P-1} x_i\right)^2}{P \cdot \sum_{i=0}^{P-1} x_i^2}, \tag{12}$$

where $x_i$ is a measure of the shared resource at node $i$.

In particular, the quantity we consider is the number of data points of cluster $k$ at node $i$, or equivalently the probability defined in (11). The distribution of the data points of cluster $k$ over the network is measured by a cluster Jain index $J(k)$:

$$J(k) = \frac{\left(\sum_{i=0}^{P-1} Prob(i,k)\right)^2}{P \cdot \sum_{i=0}^{P-1} Prob(i,k)^2}. \tag{13}$$

The overall Jain index for a data distribution is the average cluster Jain index:

$$J = \frac{1}{K}\sum_{k=1}^{K} J(k). \tag{14}$$

The Jain index is continuous and bounded in the range $[0, 1]$. It is independent of the scale, the metric and the total amount of the shared resource and of the number of contending entities. In our context, the index is a measure of the cluster data distribution over the network nodes. An index closer to 1 means a better fairness in the allocation, i.e. a uniform distribution. An index closer to 0 means a skewed allocation, i.e. a non-uniform distribution with cluster data concentrated around the topological cluster centres.

Figure 7 shows the cluster distributions for four values of the parameter $\lambda$ $(0, 2, 3, 5)$ for a 2D mesh topology. Each cluster is associated with a colour. The coloured diagram is a matrix representing the network topology. Each cell of the matrix corresponds to a network node and contains pixels of various colour corresponding to its local data points. In Figure 7(a) there is no emerging colour pattern as clusters are uniformly distributed over the network ($J \approx 1$). On the contrary in Figure 7(d) the colour patches around the topological cluster centres are clearly visible ($J \approx 0$).

### *7.2.2. Initialisation and termination condition*

All algorithms have been executed under the same conditions: the same data and cluster distributions and the same initial centroids. Two different sets of initial centroids were randomly generated for each test case.

The sequential algorithm was run till full convergence. The total number of iterations varied for the different configurations roughly in the range $[20, 80]$. On average the sequential algorithm achieved full convergence in 43 iterations.

In each distributed algorithm each node performs a constant number of communication operations at each iteration. The Aggregation protocol in *Epidemic K-Means* on average sends 2 messages at each Gossip cycle ($NC = 15$) in each K-Means iteration. *Random P2P K-Means* was run with two different parameter values for the number of neighbours: 4 as used in [15] and 15 to match the number of messages sent by *Epidemic K-Means*. Both *Random P2P K-Means (*15*)* and *Epidemic K-Means* send 30 messages at each K-Means iteration.

The number of iterations required to converge is important as it determines the total number of messages. Each distributed algorithm has its own early termination condition based on the same threshold value $\tau = 1\%$.

The *Epidemic K-Means* algorithm adopts the relative variation of the global objective function (2). In all simulations the algorithm has always converged within a small number of iterations (at most 8).

In *Local P2P K-Means* each node converges when the local updates of the centroids are below the threshold [14, 17]. On average it converged in 14 iterations. In some simulations the algorithm has converged in a number of iterations ($> 80$) greater than the centralised algorithm.

*Random P2P K-Means* adopts a similar approach to the *Local P2P K-Means* with the difference that the convergence criterion must be verified at the random communication partners [15]. On average it has converged in 87 iterations.

*Local P2P K-Means* did not converge spontaneously in a few cases, where the execution was forced to terminate after 100 iterations. *Random P2P K-Means* did not converge in 95% of the simulations, where the execution was forced to terminate after 100 iterations. In such cases *Local P2P K-Means Random P2P K-Means* have shown an oscillating behaviour because of the highly skewed cluster distribution.

### *7.2.3. Comparative Analysis*

We have compared the three distributed approaches by computing their performance with respect to two main criteria: how well each algorithm approximates the *ideal* K-Means solution and how similar the solutions at different nodes are.

A total of 168 simulations have been carried out varying the cluster distribution, the topology type, the network size and the initial centroids. The results have been averaged and are presented in function of the cluster distribution (Jain index).

Figure 8 shows the average cluster accuracy with respect to the *ideal* solution and its standard deviation over the network nodes according to (8). The *Epidemic K-Means* always outperforms the other two approaches, even for uniformly cluster distributions. Moreover, the performance of the proposed approach is constant and the standard deviation is very low across the entire range of the Jain index.

Figure 8(a) indicates that the error of the proposed approach is very small, but not null. The reason is that the distributed algorithm is using the early termination criterion based on a minimum relative improvement, while the sequential K-Means algorithm is run till full convergence.

Figure 9(a) shows the mean squared error of the final centroids w.r.t. the *ideal* solution according to (5). The error of the proposed approach is very low. The two sampling-based algorithms always show a significant error. For uniformly distributed cluster data their error is relatively low, while for more skewed data distributions their error becomes larger. The local approach obviously performs even worse than the random approach when clusters are more localised.

Figure 9(b) shows the mean squared error of the final centroids at different network nodes (the consistency error) according to (6). This chart confirms that only the proposed approach can achieve an important objective: the final clustering is strictly consistent in all network nodes (the line overlaps with the x axis).

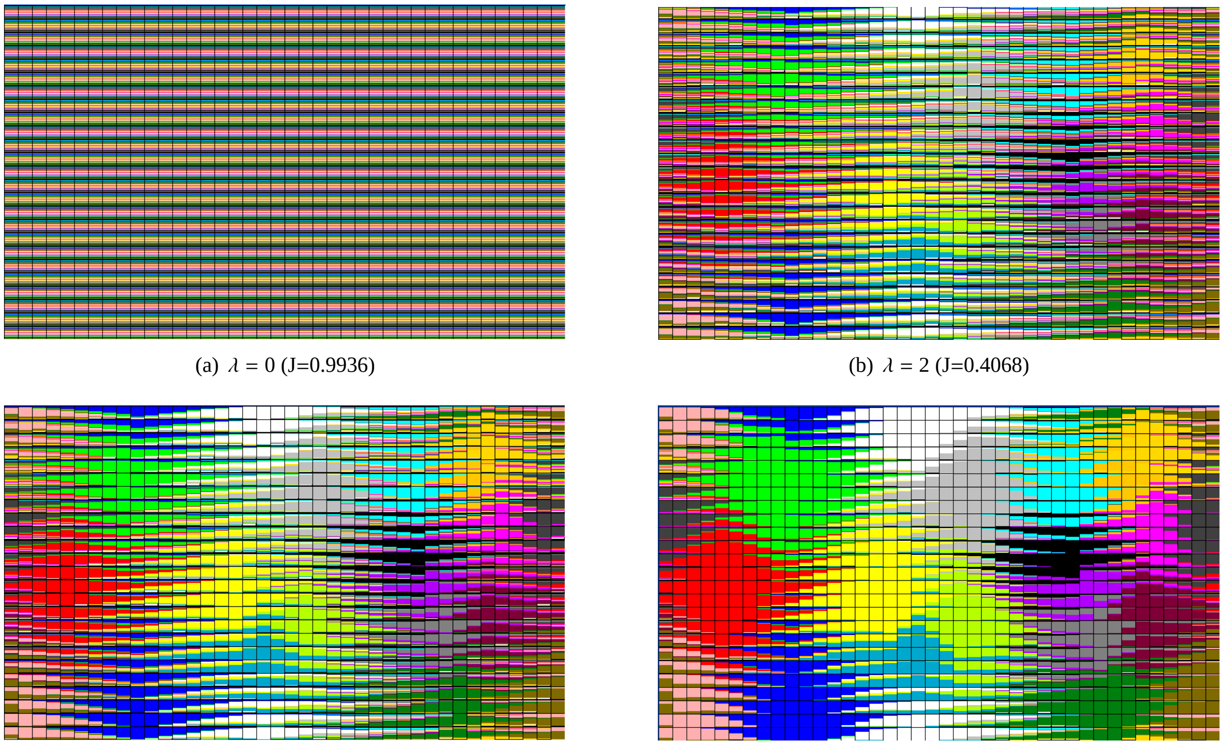

(a) $\lambda = 0$ (J=0.9936) (b) $\lambda = 2$ (J=0.4068) (c) $\lambda = 3$ (J=0.2297) (d) $\lambda = 5$ (J=0.1181)

Figure 7: Four examples of cluster distributions

The experimental analysis has shown that the proposed approach outperforms the sampling-based approaches for all data distributions, even for uniformly cluster distributions. As expected, for skewed data distributions the performance gap becomes more significant. Most notably, *Local P2P K-Means* and *Random P2P K-Means* find different clustering solutions than the *ideal* sequential K-Means and find different solutions at different nodes. The proposed approach provides a very good approximation of the *ideal* solution and the same solution is obtained by all network nodes.

### 7.3. *Epidemic K-Means Fault Tolerance*

Finally we have investigated the fault tolerance of the *Epidemic K-Means* algorithm to message losses and node failures.

Message losses may be disregarded when using a reliable transport protocol, such as TCP. However, the packet retransmission mechanism in TCP can introduce unbounded delays which may make messages obsolete (related to a past Gossip cycle) and consequently discarded at the application layer. In both cases of reliable and unreliable transport protocols, packet loss and discarded messages introduce a violation of the mass conservation property and represent a source of inaccuracy for the Gossip-based Aggregation protocol. An unreliable transport protocol is used and nodes do not detect the lost messages. During each simulation messages are randomly discarded with a uniform probability distribution.

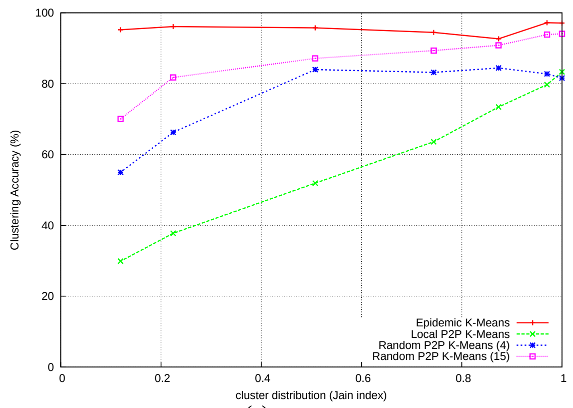
We have adopted a lifetime-based node failure model [36] with a stationary overlay network size, i.e. for each node leaving the system a node joins in. Nodes' lifetime follows a shifted Pareto distribution $F(x) = 1-(1+\frac{x}{\beta})^{-\alpha}$, where $x > 0$ and $\alpha > 1$. In the simulations we adopted a Pareto index $\alpha = 3$ and varied the scaling factor $\beta \in [7', 240']$, which corresponds to an average lifetime in the range between 210" and 2h. Churn is the rate at which nodes join and leave the system. The above configuration generates churn rates in the range between 1% and 20%.

We have adopted a stationary data distribution; data distribution at nodes does not change over time and follows the same approach described in Section 7.2.1, from uniform to skewed distributions.

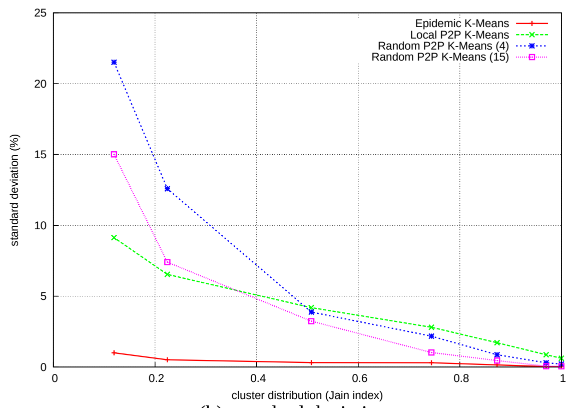
We assume that nodes take random departure decisions to simulate sudden node failures; nodes do not perform any particular procedure when leaving the system, nor node failures are detected. As a consequence, messages sent to departed nodes are lost. Similarly when a node joins the system, it simply recovers computation and communication from the previous local state and, if not available, from the initial configuration (i.e. random centroids).

Simulations were carried out to investigate the performance degradation of the *Epidemic K-Means* algorithm with a varying percentage $(0-20\%)$ of uniform message loss and node churn.

Figure 10 shows the average Clustering Accuracy and its standard deviation over the network nodes for both types of failures. Figure 11 shows the mean squared error over the *ideal* centroids ($MSE_{cKM}$) for the simulations for both types of failures. The consistency error among network nodes is not shown: it is not effected by network failures and is always close to zero,

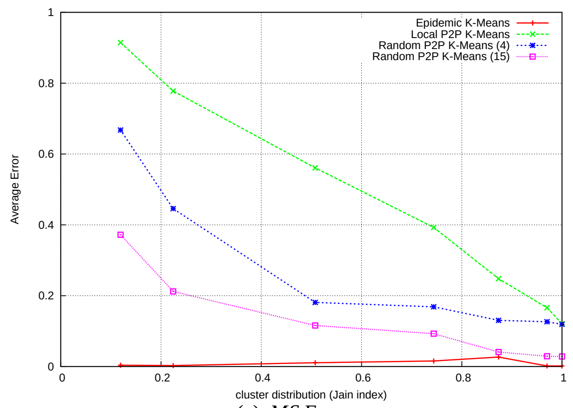


(a) average

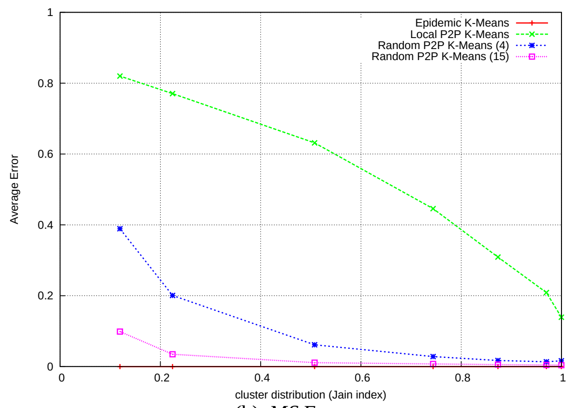


(b) standard deviation

Figure 8: Clustering accuracy at network nodes



(a) $MSE_{cKM}$



(b) $MSE_{dKM}$

Figure 9: Mean squared error of the cluster centroids w.r.t. the *ideal* centroids ($MSE_{cKM}$) and the consistency error among network nodes ($MSE_{dKM}$)

similarly to Figure 9(b).

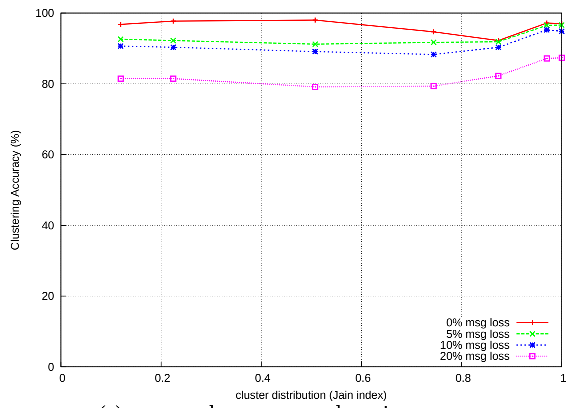
All cases confirm the intrinsic fault tolerant characteristics of the *Epidemic K-Means* algorithm. The performance degrades gracefully under higher rates of network failures.

Some conclusive remarks can be drawn by looking at the worst simulation scenario. For highly skewed data ($J = 0.12$) and with a high rates of network failures (20% of either message losses or node churn), *Epidemic K-Means* is still performing better than any of the sample-based approaches in faultless network conditions (see Figure 8(a)): it provides a cluster accuracy of 81% against 30% and 70%, respectively, for *Local P2P K-Means* and *Random P2P K-Means* with 15 neighbours.

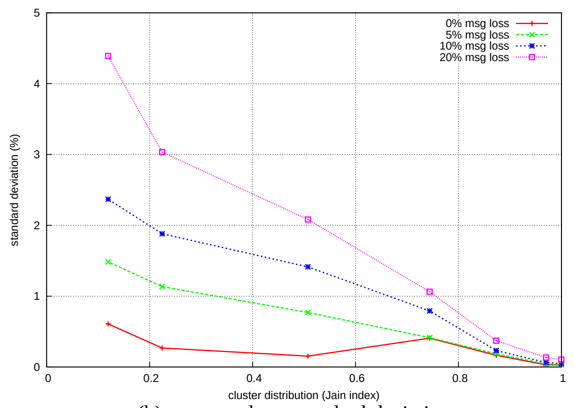
In all simulations the assumption of stationary network size and stationary data distribution is necessary to generate an ideal K-Means clustering solution against which comparing the solution obtained by the distributed algorithm at convergence. The distributed algorithm could be run in continuous mode to adapt to non-stationary data distributions. However, in non-stationary conditions there is not a single ideal K-Means clustering solution and the performance analysis would be difficult and arguable. In this case, convergence could only be achieved when the system persists in stationary conditions long enough, i.e. for a sufficient number of K-Means iterations. As reported in Section 7.2.2, the convergence of the proposed approach is achieved in a very low number of iterations ($< 10$) on average; while the approaches based on a sampling strategy, such as *Random P2P K-Means*, do not converge with non-uniform data distributions and have been terminated in 100 iterations in the 95% of the simulations. Moreover, these methods cannot naturally cope with failures as they are based on deterministic communication patterns.

## 8. Conclusions

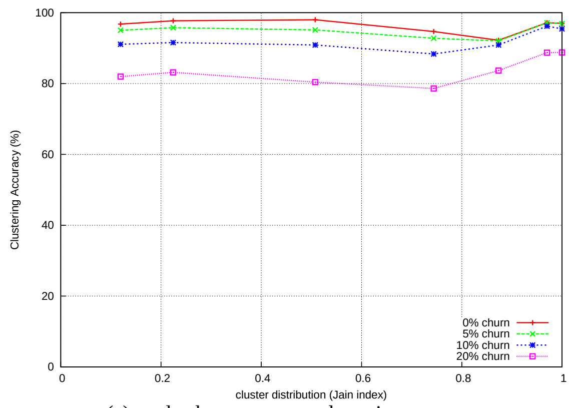
We have presented *Epidemic K-Means*, a formulation of the distributed K-Means clustering algorithm which is suitable for asynchronous networked systems of any scale. The proposed algorithm is completely decentralised and intrinsically fault tolerant. The global synchronisation and consistency required in the K-Means algorithm is achieved by means of a Gossip-based aggregation protocol. The experimental analysis has confirmed the convergence to the clustering results which would be obtained by a monolithic K-Means approach over the aggregated data. The statistical guarantees of the Gossip protocol ensure



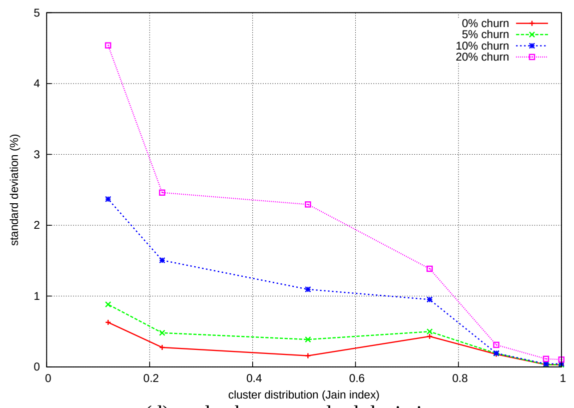
(a) message loss: average clustering accuracy



(b) message loss: standard deviation



(c) node churn: average clustering accuracy



(d) node churn: standard deviation

Figure 10: Clustering accuracy at network nodes with failures

that the results are consistent within a bounded approximation error at every node of the system in static and faultless networks. The results from extensive simulations have shown that the proposed approach always performs better than the state of the art distributed K-Means algorithms for large scale systems. The simulations have also shown that its performance degrades gracefully under message loss and node failures in asynchronous networks.

*Epidemic K-Means* is the first complex data mining algorithm which is based on Gossip protocols. A similar approach can be adopted to implement other data mining algorithms for distributed systems of large and extreme scale. Their adoption in ubiquitous computing and networking may open up a new range of powerful applications which have not been possible so far.

An adaptive termination condition may need to be devised when the approximation error of the global aggregation is greater than the termination threshold. In this case nodes may not terminate synchronously. If and how a Gossip protocol under adversarial failures may still be used to guarantee a synchronous termination of all nodes with high probability, is an interesting open issue.

## References

[1] M. Weiser, Hot topics-ubiquitous computing, Computer 26 (10) (1993) 71 –72.

[2] A. Bonifati, A. Cuzzocrea, Efficient fragmentation of large XML documents, in: Proceedings of the 18th international conference on Database and Expert Systems Applications, DEXA '07, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 539–550.

[3] A. Cuzzocrea, F. Furfaro, D. Saccà, Enabling OLAP in mobile environments via intelligent data cube compression techniques, Journal of Intelligent Information Systems 33 (2) (2009) 95–143.

[4] M. Inaba, N. Katoh, H. Imai, Applications of weighted Voronoi diagrams and randomization to variance-based k-clustering, in: SCG '94: Proceedings of the tenth annual symposium on Computational geometry, ACM, New York, NY, USA, 1994, pp. 332–339.

[5] D. Aloise, A. Deshpande, P. Hansen, P. Popat, Np-hardness of euclidean sum-of-squares clustering, Machine Learning 75 (2009) 245–248.

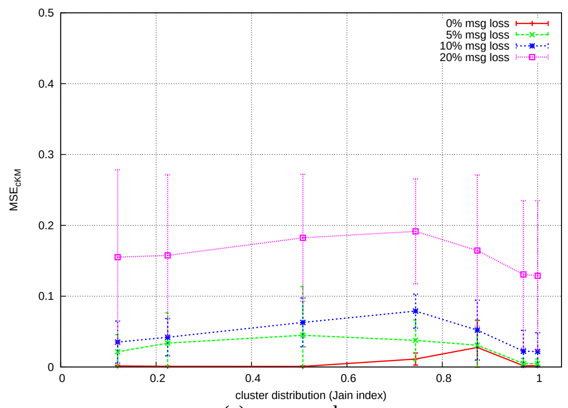
[6] J. B. MacQueen, Some methods for classification and analysis of multivariate observations, in: L. M. L. Cam, J. Neyman (Eds.), Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability, Vol. 1, University of California Press, 1967, pp. 281–297.

[7] S. P. Lloyd, Least squares quantization in PCM, IEEE Transactions on Information Theory IT-28 (2) (1982) 129–137.

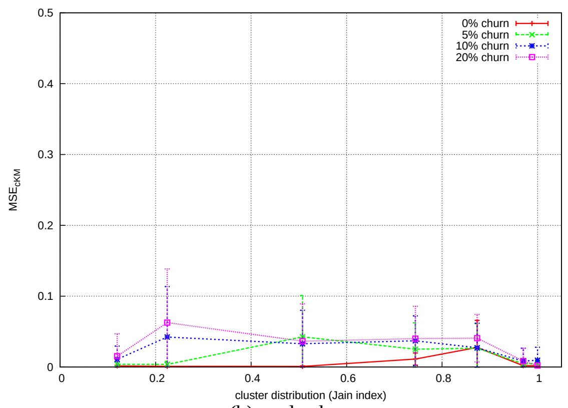
[8] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. McLachlan, A. Ng, B. Liu, P. Yu, Z.-H. Zhou, M. Steinbach, D. Hand, D. Steinberg, Top 10 algorithms in data mining, Knowledge and Information Systems 14 (2008) 1–37.

[9] A. K. Jain, Data clustering: 50 years beyond k-means, Pattern Recognition Letters 31 (8) (2010) 651 – 666, award winning papers from the 19th International Conference on Pattern Recognition (ICPR), 19th International Conference in Pattern Recognition (ICPR).

[10] D. Kempe, A. Dobra, J. Gehrke, Gossip-based computation of aggregate



(a) message loss



(b) node churn

Figure 11: Mean squared error of the cluster centroids w.r.t. the *ideal* centroids ($MSE_{\epsilon KM}$) under failures

information, in: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science, 2003, pp. 482 – 491.

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

[12] S. Boyd, A. Ghosh, B. Prabhakar, D. Shah, Randomized gossip algorithms, Information Theory, IEEE Transactions on 52 (6) (2006) 2508 – 2530.

[13] S. Datta, C. Giannella, H. Kargupta, K-means clustering over peer-to-peer networks, in: Proceedings of the 8th International Workshop on High Performance and Distributed Mining (HPDM'05). In conjunction with the SIAM International Conference on Data Mining, Newport Beach, California, 2005.

[14] S. Datta, C. Giannella, H. Kargupta, K-means clustering over a large, dynamic network, in: Proceedings of the Sixth SIAM International Conference on Data Mining, Bethesda, Maryland, USA, 2006, pp. 153–164.

[15] S. Bandyopadhyay, C. Giannella, U. Maulik, H. Kargupta, K. Liu, S. Datta, Clustering distributed data streams in peer-to-peer environments, Information Sciences, Elsevier, 176 (14) (2006) 1952 – 1985.

[16] S. Datta, K. Bhaduri, C. Giannella, R. Wolff, H. Kargupta, Distributed data mining in peer-to-peer networks, Internet Computing, IEEE, 10 (4) (2006) 18 –26.

[17] S. Datta, C. Giannella, H. Kargupta, Approximate distributed k-means clustering over a peer-to-peer network, IEEE Transactions on Knowledge and Data Engineering 21 (10) (2009) 1372–1388.

[18] D. Foti and D. Lipari and C. Pizzuti and D. Talia, Scalable parallel clustering for data mining on multicomputers, Proceedings of the 15th IPDPS 2000 Workshops on Parallel and Distributed Processing, Lecture Notes in Computer Science (2000) 390–398.

[19] D. Judd and P. K. McKinley and A. K. Jain, Large-scale parallel data clustering, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (8) (1998) 871–876.

[20] I. S. Dhillon, D. S. Modha, A data-clustering algorithm on distributed memory multiprocessors, In Large-Scale Parallel Data Mining, Lecture Notes in Computer Science 1759 (2000) 245–260.

[21] M. P. I. Forum, MPI: A message-passing interface standard, version 2.2, mpi22-report.pdf (Sep. 2009).
URL `http://www.mpi-forum.org/docs/mpi-2.2/`

[22] D. Pettinger, G. D. Fatta, Scalability of efficient parallel K-Means, in: Proceedings of the 5th IEEE International Conference on e-Science, Workshop on Computational e-Science, 2009, pp. 96–101.

[23] G. D. Fatta, D. Pettinger, Dynamic load balancing in parallel KD-Tree K-Means, in: Proceedings of the International IEEE Conference on Scalable Computing and Communications (ScalCom), 2010, pp. 2478–2485.

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

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

[26] P. Jesus, C. Baquero, P. Almeida, Dependability in aggregation by averaging, in: 1st Symposium on Informatics (INForum 2009), 2009, pp. 482 – 491.

[27] I. Rao, A. Harwood, S. Karunasekera, Gossip-based asynchronous and robust aggregation protocol - a pessimistic approach, in: Consumer Communications and Networking Conference (CCNC), 2011 IEEE, 2011, pp. 543 –548.

[28] J. Tsitsiklis, D. Bertsekas, M. Athans, Distributed asynchronous deterministic and stochastic gradient optimization algorithms, Automatic Control, IEEE Transactions on 31 (9) (1986) 803 – 812.

[29] E. Ogston, S. Jarvis, Peer sampling with improved accuracy, Peer-to-Peer Networking and Applications 2 (2009) 24–36.

[30] D. Gillman, A chernoff bound for random walks on expander graphs, SIAM Journal on Computing (Society for Industrial and Applied Mathematics) 27 (4) (1998) 12031220.

[31] G. Fortino, C. Mastroianni, W. Russo, A hierarchical control protocol for group-oriented playbacks supported by content distribution networks, J. Network and Computer Applications 32 (1) (2009) 135–157.

[32] A. Medina, I. Matta, J. Byers, On the origin of power laws in internet topologies, SIGCOMM Comput. Commun. Rev. 30 (2000) 18–28.

[33] D. L. Mills, On the accuracy and stability of clocks synchronized by the network time protocol in the Internet system, ACM Computer Communication Review 20 (1990) 65–75.

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

[35] R. Jain, D. Chiu, W. Hawe, A quantitative measure of fairness and discrimination for resource allocation in shared computer systems, Tech. rep., DEC Research Report TR-301 (1984).

[36] D. Leonard, Z. Yao, V. Rai, D. Loguinov, On lifetime-based node failure and stochastic resilience of decentralized peer-to-peer networks, IEEE/ACM Transactions on Networking 15 (3) (2007) 644 –656.