Highly Dynamic Distributed Computing with Byzantine Failures

Rachid Guerraoui
Lab. de Programmation Distribuée
EPFL, Switzerland
rachid.guerraoui@epfl.ch

Florian Huc
Lab. de Programmation Distribuée
EPFL, Switzerland
florian.huc@gmail.com

Anne-Marie Kermarrec
INRIA Rennes Bretagne-Atlantique
France
anne-marie.kermarrec.@inria.fr

ABSTRACT

This paper shows for the first time that distributed computing can be both reliable and efficient in an environment that is both highly dynamic and hostile. More specifically, we show how to maintain clusters of size $O(\log N)$, each containing more than two thirds of honest nodes with high probability, within a system whose size can vary *polynomially* with respect to its initial size. Furthermore, the communication cost induced by each node arrival or departure is polylogarithmic with respect to N, the maximal size of the system. Our clustering can be achieved despite the presence of a Byzantine adversary controlling a fraction $\tau \leq \frac{1}{3} - \epsilon$ of the nodes, for some fixed constant $\epsilon > 0$, independent of N. So far, such a clustering could only be performed for systems whose size can vary constantly and it was not clear whether that was at all possible for polynomial variances.

Categories and Subject Descriptors

F.2.2 [**Theory of Computation**]: Analysis of algorithms and problem complexity—*Nonnumerical Algorithms and Problems*

Keywords: Byzantine failures ; random walks ; dynamic networks

1. INTRODUCTION

Distributed computing can be achieved reliably in a system where at most one third of the processes are controlled by an adversary. Typically, assuming some synchrony, the seminal agreement problem [25] can be solved and used to emulate a single highly available process. This is a basic building block to achieve distributed computations in a reliable manner. Yet, with a large number of nodes, this technique is very expensive. One way to reduce the complexity consists in clustering the nodes within smaller subsets, picked randomly, so that each cluster contains two third of correct nodes whp, e.g., as proposed in [11]. In short, instead of reducing a system of many processes into a system of one reliable process that performs the computation, the idea here is to reduce it to a system of several reliable processes, each corresponding to one of the clusters. These processes share the load of the computations reducing thereby their complexity.

So far, clustering techniques mainly assumed a static distributed system: the number n of processes is assumed to be fixed a priori and processes do not join or leave the system [14] (a few can typically fail). Some approaches have explored dynamic settings, but in a limited fashion: the number of processes n is assumed to only vary by a constant factor [6, 7, 12, 31]. Yet, whether this is at all possible to go beyond has been considered an open question so far [18, 19].

This paper answers the question positively. We show, for the first time, that it is possible to perform distributed computing reliably and efficiently in a system which size can vary in a *polynomial* manner. At the heart of this result lies a new technique to partition nodes in a dynamic number of clusters, which involves a radical departure from previous schemes that assume a static number of clusters [6, 7, 12, 31]. Indeed, tolerating an increase in the number of nodes from n to n^2 (and more generally from $n^{1/y}$ to n^z for some constants $y, z > 1$), with a static number of clusters, yields a significant increase in the number of nodes within each cluster, leading to a high-complexity computation, in the vein of a single cluster approach. However, handling dynamic clusters is not trivial. For instance, using classical De Brujin graphs for clustering [6] in a dynamic setting requires a good estimation of the number of nodes. In turn, this potentially requires techniques with high complexity, e.g., typically $\tilde{O}(n^{3/2})$ [24].

Our clustering approach achieves a polylogarithmic complexity by using random walks on expander graphs with small degrees. To ensure that each cluster contains two thirds of correct nodes with high probability, we exchange nodes between clusters whenever new nodes join or leave the system. The nodes that are candidate to the exchange are selected using continuous random walks [1]. These provide a uniformly chosen sample even if the underlying graph is not regular. To ensure that a walk ends up fast on a node picked quasi uniformly, we connect clusters through small degree expanders [20].

The distributed construction of this expander requires specific care in regulating the choice of edges. Although

PODC'13, July 22–24, 2013, Montréal, Québec, Canada.

several expanders could be used, our approach relies on OVER, a technique (Over-Valued Erdös Rèiny graph) from Erdös Rèiny random graphs to preserve a small degree and a good expansion coefficient. OVER is described in the long version of this paper [16] for space reasons. This technique tolerates more crashes than [3, 15, 26] and yields a different degree than [2]. In the rest of the paper, we present NOW (Neighbors On Watch), a protocol maintaining the cluster partition in Section 3 and analyze it in Section 4. We review the related work in Section 5 and conclude in Section 6.

2. MODEL AND BACKGROUND

System assumptions.

In short, our network model is the one of [7] with the difference that we allow the size of the system to vary *polynomially*. More specifically, we consider a dynamic synchronous network with a discrete time variable t_i. Each node can send messages to any node it knows through a private channel; in this sense the network is reconfigurable as connections between nodes can be added or removed. We do not assume that each node knows all other nodes in the network (except during the initialization phase in which the global structure of the network is computed once). Instead, each node knows polylog(N) nodes and only knows an upper bound on the current size of the network. We also assume that, initially, the number of nodes is n_{t_0} for some $\sqrt{N} \leq n_{t_0} \leq N$, and the current number of nodes n in the network remains between $\sqrt{N}$ and N (this can be relaxed to $N^{1/y} \leq n \leq N^z$ for all constants $y, z > 1$). The size of the network can increase or decrease at any time. For simplicity of presentation, we assume (as in [7, 26]) that when a node joins or leaves, the actions relative to previous joins and leaves are over. This corresponds to a time step.ËIJ* Moreover, nodes do not need to take any specific action when leaving the network. Instead, we assume a mechanism enabling a node to detect if one of its neighbors has crashed or left the network.

Adversary model.

Our adversary is that of [7] with the difference that in our case it controls a fraction $\tau \leq \frac{1}{3} - \epsilon$ (for some constant $\epsilon > 0$) nodes, from the beginning (*vs.* $\tau \leq \frac{1}{2} - \epsilon$ after some initialization phase; note that using cryptographic tools, we could also assume $\tau \leq \frac{1}{2} - \epsilon$ by leveraging broadcast algorithms [13].).

NOW tolerates a static Byzantine (sometimes called *active*) adversary controlling a fraction $\tau \leq \frac{1}{3} - \epsilon$ (for some constant $\epsilon > 0$) of the nodes, having a full knowledge of the network at any time, as in [7, 18], i.e it knows the position of any node at any time. A typical objective for the adversary is to gain the lead in one (or more) of the clusters. At the beginning of the protocol, the adversary can choose a fraction τ of the nodes to corrupt. We assume that, at initialization, the honest nodes form a connected component, that the adversary cannot split it into disjoint parts, that each node controlled by the adversary is adjacent to at least one honest node, and that no honest node leaves or joins the network until the initialization is over. Also, nodes' identities cannot be forged. Moreover during the execution of the protocol, each time a node joins the network, the adversary can choose to corrupt it or not, as in [6, 7]. However, the adversary cannot decide to corrupt nodes at a later time (in this sense the adversary is static and not adaptive). Furthermore, the adversary can induce churn as in [6, 7] by join-leave attacks or by forcing honest nodes to leave the system (e.g., through a DOS attack). The size of the network can vary polynomially and each node is assigned a unique identifier.

Background on OVER: expander graph.

Our clustering technique, which we call NOW (Neighbors On Watch) and that maintains a cluster partition is based on a protocol to distributely maintain an expander overlay. Although various expanders (e.g. [2]) could be used, we assume that NOW relies on OVER. For space reasons, the detailed description of OVER is deferred to the long version of the paper [16]. In OVER, the graph vertices represent the clusters of nodes maintained by NOW, hence they can be considered as honest since each vertex is composed of more than two thirds of honest nodes whp. We further assume that each vertex leaving the overlay graph is chosen at random (this assumption will be ensured in Section 3.3).

OVER ensures that, starting from a random graph drawn from the Erdös-Rényi model, whp, at any time during a sequence of vertex additions and removals polynomial in N, the resulting graph has a large isoperimetric constant and a low degree (ensuring properties 1 and 2). We use the notation $\widehat{G}^R$ where the $\hat{}$ relates to the fact that we consider an overlay, and the ^{R} that it is an instance of a random graph. The evolution of the graph is represented by a sequence $\widehat{G}^R_{t_0}, \ldots, \widehat{G}^R_{t_i}, \ldots$. n_{t_i} denotes the number of vertices of $\widehat{G}^R_{t_i}$.

PROPERTY 1. *Whp, at any time t after a number of time steps polynomial in n, $\widehat{G}^R_t = (\hat{V}^R_t, \hat{E}^R_t)$ has an isoperimetric constant $I(\widehat{G}^R_t) \geq \log^{1+\alpha} N/2$, where:*
$I(\widehat{G}^R_t) = \inf_{S \subset \hat{V}^R_t : |S| \leq n_t/2} E(S, \bar{S})/|S|$.

PROPERTY 2 (MAXIMUM DEGREE OF $\widehat{G}^R$). *Whp, at any time t after a number of time steps polynomial in n, $\widehat{G}^R_t$ has maximal degree at most $c \log^{1+\alpha} N$ for a large enough constant c and an arbitrarily small (pre-)chosen constant α.*

Those properties enable to achieve short random walks leading to pick nodes uniformly at random. Note that OVER enables to tolerate simultaneous failures as long as the targets are picked uniformly at random. NOW together with OVER can also tolerate the failures of nodes chosen by the adversary as long as one failure per round is assumed.

Notations.

We use the time step as a subscript to indicate the instant at which a variable is considered (e.g., n_{t_i} is the number of nodes at time t_i, $\#C_{t_i}$ is the number of clusters, and $|C_j|_{t_i}$ the size of C_j). We may omit the index of the time

9* However, the analysis can be generalized to several parallel join and leave operations.

step when there is no ambiguity (e.g., n stands for the current number of nodes in the network). The *communication cost* is the number of messages[†] exchanged, and the *round complexity*, is the number of communication rounds (i.e. the number of successive messages) required by a protocol to terminate. Notice that a time step is composed of several communication rounds, but we will prove that they are polylog(N). Given a graph $G = (V, E)$, and a vertex $v \in G$, we denote by d_v its degree. Similarly, for a given cluster C, d_C denotes the number of clusters adjacent to C.

3. NOW: OVERLAY OF CLUSTERS

NOW (Neighbors On Watch) maintains both an overlay of clusters and the partition of the nodes into clusters. NOW relies on the fact that the overlay is guaranteed to have a low maximum degree and good expansion properties. This is provided by the protocol OVER that we present in the long version of the paper [16] but could also be ensured by other protocols which differ either in the number of failures they can provide [3, 15, 26] or their degree (e.g., 4 in [2] instead of $\log^{1+\alpha} N$ in OVER for some arbitrarily small constant $\alpha > 0$)). NOW further ensures that each cluster contains more than two thirds of honest nodes whp. The clusters have size $O(\log N)$ and are used to inhibit the behavior of the Byzantine nodes. NOW relies on two phases: *initialization* and *maintenance*. In a nutshell, the initialization phase generates the initial overlay, while the maintenance phase ensures that after a polynomially long sequence of leave and join operations, the required properties still hold. The overlay $\widehat{G}^R$ is first constructed during the initialization phase of NOW, and recursively maintained by OVER as described in [16].

3.1 Preliminaries

A node of a cluster C is linked to all the other nodes of C and knows their identities. An edge between two clusters C_i and C_j in $\widehat{G}^R$ means that all nodes of C_i are linked to all nodes of C_j and know their identities (and *vice-versa*). A node only needs to know the identities of the nodes in its cluster and the neighboring ones. The initialization phase (Section 3.2) is itself divided into two sub-phases. First, a discovery algorithm is run in order for the nodes to acquire a global knowledge of the network. Afterwards, a Byzantine agreement algorithm [19] is used to construct an initial overlay of clusters. The maintenance phase ensures that each cluster contains more than two thirds of honest nodes whp when nodes join or leave and preserves the properties of the overlay.

Random number generation.

We assume the existence of `randNum`, a distributed random number generation protocol, enabling the nodes of a cluster to agree on a common integer chosen uniformly at random from the interval $(0, r)$. `randNum` is secure as long as the Byzantine nodes are less than two thirds in the cluster and is presented the long version of the paper [16].

Cluster random choice.

Furthermore, we assume the existence of a function called `randCl` (Algorithm in [16]), to randomly select a cluster. To achieve the random selection (`randCl`), we perform a biased CTRW [‡] on $\widehat{G}^R$, the overlay. We bias our CTRW such that a cluster is chosen according to the distribution ($|C_i|/n$). With clusters of size $O(\log N)$, this primitive has an expected communication cost of $O(\log^5 N)$. Indeed, the expected number of clusters visited during the walk is $O(\log^3 N)$ (whp, we do $O(\log n)$ CTRW each of length $O(\log^2 n)$) and at each cluster a random integer from the range $(0, O(\log^{1+\alpha} N))$ is generated at a cost of $O(\log^2 N)$. The expected round complexity of this primitive is $O(\log^4 N)$.

Node shuffling.

In order to avoid an adversary to focus on one cluster and gradually pollute it with Byzantine nodes, shuffling nodes between clusters is necessary upon nodes arrival and departure. The shuffling is implemented by the algorithm called `exchange` and detailed in [16]. Basically some clusters exchange their nodes with nodes chosen at random from other clusters. For each node x to be exchanged from cluster C (x is determined by the protocol `exchange`), a cluster is chosen at random using `randCl`. The chosen cluster, C', is informed that it will receive x. The cluster C' chooses one of its nodes (using the primitive `randNum`) to send in replacement of x. During an exchange, if C is adjacent to another cluster, the nodes of this cluster are informed of the new composition of C. This step is fundamental since a node from a neighboring cluster accepts a message from C if and only if at least half plus one of the nodes of C send it. The new nodes of C are informed by the former nodes of this cluster of the local structure of the overlay (i.e., the neighboring clusters of C in the overlay). The expected communication cost and round complexity of `exchange` are $O(\log^6 N)$ and $O(\log^4 N)$.

3.2 NOW: Initialization Phase

Network Discovery.

The protocol starts by running an algorithm that informs each node of the identifiers of all other nodes. The global knowledge of the nodes in the network is needed only at initialization. Note that this computation is performed while the size of the network is still "small" in practice. Afterwards, it is possible to use standard off-the-shelf Byzantine agreement algorithms to construct an initial partition forming the vertices of the overlay $\widehat{G}^R$. This algorithm (Provided in [16]) terminates after a number of communication rounds at most the diameter of the graph considering only the edges adjacent to at least one honest node. When the algorithm terminates, it is guaranteed that all

9[†] We consider messages of identical size. Hence the communication cost is proportional to the number of bits sent.

9[‡] A vertex C_i of $\widehat{G}^R$ is a cluster in G. A biased CTRW from C_i is a sequence of CTRW as follows: the nodes of C_i choose collaboratively the next cluster C_j and decrease the duration of the CTRW using `randNum` which goes on similarly. When the remaining duration is negative or null, a random number between 0 and 1 is chosen. If it is smaller than $|C_i|/\max_C |C|$, the biased CTRW ends, otherwise a CTRW starts again. A node of a cluster C_j pursues the random walk if and only if it receives an identical message from at least half plus one of the nodes of the neighboring cluster from which the CTRW comes.

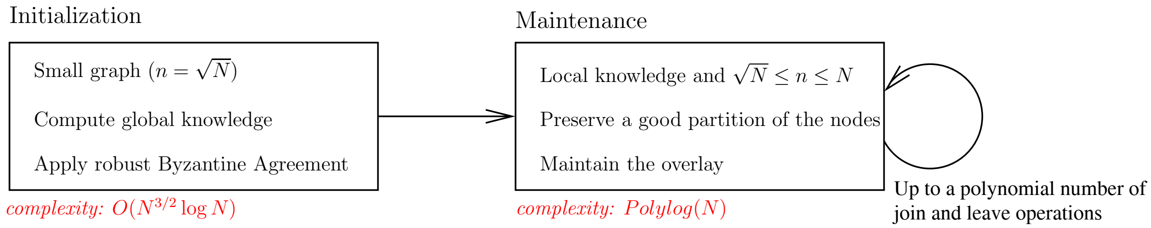

Figure 1: Overview of NOW.

honest nodes know the identities of all nodes in the network. Its communication cost is $O(n \times e)$ where $e = |E|$ (see [16] for the theorem and details).

Clusterization.

Once all the honest nodes know the identities of all the nodes in the network, any Byzantine agreement protocol can be used, such as [19] whose complexity is $\tilde{O}(n\sqrt{n})$. This protocol works in the presence of a static Byzantine adversary controlling less than $1/3 - \epsilon$ of the nodes for some positive constant ϵ. A representative cluster of logarithmic size containing more than two thirds of honest nodes is selected. Afterwards, we use the nodes of this representative cluster to randomly partition the network into $\#C$ clusters, $\{C_1, \ldots, C_{\#C}\}$, each of size $k \log N$, for some constant k. The constant k is a security parameter of the protocol that is chosen *a priori* depending on the requirements of the application considered: the higher k, the less chances the adversary has to control more than a third of the nodes of one of the clusters. Choosing the partition at random ensures that whp, there is more than two thirds of honest nodes in each cluster. This can be proved using standard Chernoff bound and union bound arguments. To obtain a random partition, it is sufficient for the representative cluster to order the nodes at random by calling the primitive `randNum`. Once the random ordering has been computed, the partition is obtained by taking for each cluster $k \log N$ successive nodes. Afterwards, $\widehat{G}^R_{t_0}$ is initiated on top of this partition: for each pair of clusters, the representative cluster determines with probability $p = \log^{1+\alpha} N/\sqrt{N}$ whether or not they will be linked by an edge in $\widehat{G}^R_{t_0}$. Finally, the representative cluster tells each node x the cluster it belongs to, the identities of the other nodes in this cluster, and the adjacent clusters as well as their composition (i.e., the identities of the nodes). The node x is "linked" to all these nodes and can for efficiency purposes forget the identifier of any other node that it may know. It is fundamental for the security of our protocol that each cluster contains more than two thirds of honest nodes. Indeed, a node receiving a message from all the nodes of a particular cluster considers this message valid if and only if, it receives the same message from more than half of the nodes of this cluster. Using this rule for inter-cluster communication, together with the condition that each cluster has more than two thirds of honest nodes, is sufficient to ensure the correctness of the protocol.

3.3 NOW: Maintenance Phase

While the initialization phase of NOW ensures the desired properties for both the overlay and the clusters, maintaining these properties under high dynamics is challenging. In this section, we describe how to preserve the property that each cluster is composed of an honest majority in the presence of nodes join and leave operations. Shuffling the network is crucial at this point as mentioned in [6, 7, 31] to avoid the adversary to control a majority of nodes in a cluster after a few steps by using a very simple strategy: the adversary chooses a specific cluster and keeps adding and removing the Byzantine nodes until they fall into that cluster. Similarly, it is crucial to introduce dynamics with shuffling if nodes are forced to leave the network by the adversary. The shuffling is generated upon *Join* and *Leave* operations. Complementary, the *Split* and *Merge* operations ensure that the clusters remain of size $\Omega(\log N)$, and that the required properties of $\widehat{G}^R$ (i.e., expansion and low maximum degree) are preserved.

The NOW following operations are invoked by the nodes upon joining, or leaving the network, or simultaneously by all the nodes of a cluster involved in a split or a merge operation.

Join.

This operation (as well as the leave operation), initiated by a node joining the network, is inspired by [6, 7, 31]. When a node x joins the network, we assume that it gets in contact with a cluster of the overlay. This cluster chooses another cluster using `randCl` in which x is inserted. The chosen cluster proceeds by inserting x and uses `exchange` for all of its nodes. This operation has a communication cost of polylog(N).

Split.

This operation is initiated simultaneously by all nodes of a cluster C if after a join operation, the size of this cluster is larger than $lk \log N$ for some fixed parameter l (l is a constant greater than $\sqrt{2}$ which influences the number of split and merge operations). Then C has to be split in two, the old and the new clusters. To this end, the nodes of C generate a random partition of C. The old cluster keeps its neighbors in $\widehat{G}^R$, whereas the new cluster is added to the overlay using `Add` as described in [16]. This procedure has a communication cost of polylog(N) and a $O(\log^4 N)$ round complexity. Recall that each node knows the exact composition of its cluster, therefore a split operation can be easily achieved.

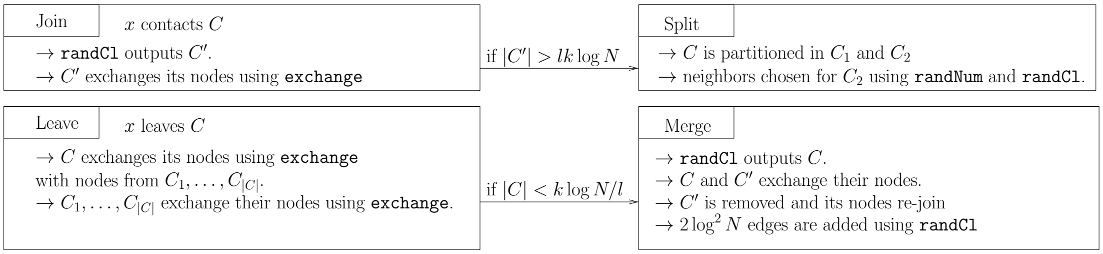

Figure 2: Maintenance of the overlay. Each operation has a $\mathrm{polylog}(N)$ **complexity.**

Algorithm 1 Join operation.

9

Require: Node x contacting cluster C to join the network.
Ensure: The preservation of the properties of the overlay and of the clusters.
Nodes of C choose a cluster C' using `randCl`.
All nodes of C' add x to their local view of C'.
All nodes of C' send a message to all the nodes from the neighboring clusters informing that x is added to C'.
All nodes of C' send their neighborhood to x using the path used to find C' in `randCl`.
if $|C'| > kl \log n$ **then**
  Nodes of C' compute a partition of C' into two parts of roughly the same size using `randCl`: C_1 and C_2.
  Nodes of C_1 keep their neighborhood.
  Nodes of C_1 and C_2 send a message informing that C' is replaced by C_1 to the neighbors of C_1.
  Nodes of C_2 are given a new neighborhood using `Add`(C_2) (Algorithmof OVER [16]).
end if

Algorithm 2 Leave operation.

9

Require: Node x from a cluster C leaving the network.
Ensure: The preservation of the properties of the overlay and the clusters.
Nodes of C remove x from their view.
Nodes of C send a message to their neighbors informing them to remove x from their view.
A node that is a neighbor of C receiving a message to remove $x \in C$ from more than half of the nodes of C removes it from its view.
C exchanges its nodes using `exchange`.
A cluster exchanging one or more of its nodes with C execute the `exchange` procedure.
if $|C'| < k \log n/l$ **then**
  Nodes of C inform all their neighbors that C is removed.
  Nodes of C execute `Remove`(C_1) ([16]) of OVER.
  A node that is a neighbor of C receiving a message that C is removed from more than half of the nodes of C removes it from its view.
  Nodes of C execute Algorithm 1 as to rejoin the network.
end if

Leave.
This operation occurs when a node from a cluster C leaves the network or when the other nodes of C detect its absence. C exchanges all its nodes using the primitive `exchange`. Then, a cluster receiving one or more nodes from C execute `exchange` for all of its nodes. This process has a communication cost of $\mathrm{polylog}(N)$ and a $O(\log^4 N)$ round complexity.

Merge.
This operation is initiated simultaneously by all nodes of a cluster C containing less than $\frac{k \log N}{l}$ users (for the same fixed parameter l described previously). In this situation, a cluster, chosen at random in order to ensure Properties 1 and 2, has to be removed. This is achieved using the primitive `randCl`. Nodes in C proceed as if they were joining the network while the nodes from the chosen cluster C' become members of C. In $\widehat{G}^R$, C' is removed by using the operation `Remove` described in [16].

4. NOW: ANALYSIS

In this section, we prove that after a polynomial sequence of join and leave operations (some of them inducing some splitting and merging of clusters), each cluster contains more than two thirds of honest nodes as long as the fraction of Byzantine nodes τ controlled by the adversary is smaller than $1/3 - \epsilon$ (for some constant $\epsilon > 0$ independent of n).

The results are proved under the assumption that the random choices of nodes are perfectly uniform (i.e, the small bias induced by the random walk is ignored). This assumption is justifed by the fact that we consider a mixing time after which the distance from the desired distribution is $O(n^{-c})$ for some arbitrarily large constant c. More specifically, we describe the output of a CTRW using two random variables X and Y. X indicates whether or not the output of the CTRW has the desired distribution and is defined as follows: we consider the probability distribution $\mathcal{D}$ of the endpoints of a CTRW, and set p_v as the probability node v is hit. Set $p_{min} = \min_v(p_v)$. The binary random variable X has value 1 with probability $n \times p_{min}$ and 0 otherwise. Y is equal to node v with probability $(p_v - p_{min})/(\sum_w (p_w - p_{min}))$. We can reproduce $\mathcal{D}$ by first evaluating X. Then, if $X = 1$, the endpoint is picked according to the desired distribution. Else, the endpoint is picked according to Y. We have $P(X = 0) \leq n \times \max(p_v - p_{min}) = O(n^{-c+1})$, which means that the probability of the endpoint not to be picked as desired is $O(n^{-c+1})$. Conditional to that, in the following we assume that the random choices made using a CTRW are as desired, i.e ($|C|/n$) for each cluster C where $|C|$ is its size.

4.1 Status of a cluster after `exchange`

At each time step, we assume that either a join or leave operation takes place or nothing occurs. These operations may in turn induce the splitting or merging of clusters. A split operation is done directly at the time it occurs, whereas, when a cluster executes a merge operation, we consider that its nodes re-join the network in subsequent time steps inducing normal join operations. Given a cluster C, p_t^C is the proportion of Byzantine nodes in C at time t.

LEMMA 1 (2/3 OF HONEST NODES IN A CLUSTER). *If a cluster C has exchanged all its nodes at time step t, we have $P(p_t^C > \tau(1+\epsilon)) \leq n^{-\gamma}$, for any positive constant γ, as long as the security parameter k is large enough.*

PROOF. When a cluster C exchanges one of its nodes with another cluster, this cluster is first selected at random according to the probability distribution $(|C|_{t_i}/n)$, and then a node is chosen out of it uniformly at random. In this scenario, the probability of performing an exchange with a Byzantine node is τ.

Using standard Chernoff bound arguments, we can derive the following result on the number X of Byzantine nodes among $|C|_{t_i}$ nodes: $P(X > (1+\epsilon)\tau|C|_{t_i}) \leq e^{-\epsilon^2\tau|C|_{t_i}/3}$. Therefore as $|C|_{t_i} \geq (k \log N)/l$, we have $P(X > (1+\epsilon)\tau|C|_{t_i}) \leq N^{-\gamma}$ when k is sufficiently large for some constant γ. □

This lemma is a consequence of the Chernoff bound arguments [17] and implies that to obtain more than two thirds of honest nodes in a cluster whp, it is sufficient that $\tau + \epsilon < 1/3$, which is true by assumption on τ.

REMARK 1 (INCREASING THE ROBUSTNESS). *One can tolerate a fraction of Byzantine nodes up to $1/2 - \epsilon$, but then we need to use cryptographic tools to allow for broadcast and Byzantine agreement.*

4.2 Evolution of the divergence

To summarize, we have seen that each time a cluster exchanges all of its nodes, as long as $\tau(1+\epsilon) < 1/3$, we obtain more than two thirds of honest nodes whp in the resulting cluster. We now proceed by proving that in between two exchanges, this property also holds. To realize this, we focus on a specific cluster C and consider a sequence of s join and leave operations.

We first prove that if the cluster has less than a $\tau(1+\epsilon/2)$ fraction of Byzantine nodes, then after it has exchanged $O(\log N)$ of its nodes, it does not have more than a $\tau(1+\epsilon)$ fraction of Byzantine nodes. Then, we prove that if it has between a $\tau(1+\epsilon/2)$ and $\tau(1+\epsilon)$ fraction of Byzantine nodes, then after it has exchanged $O(\log N)$ of its nodes, it has less than a $\tau(1+\epsilon/2)$ fraction of Byzantine nodes whp.

LEMMA 2. *If a cluster C has less than $\tau(1+\epsilon/2)|C|$ Byzantine nodes, then after $O(\log N)$ node exchanges with nodes chosen uniformly at random, the cluster does not contain more than $\tau(1+\epsilon)|C|$ Byzantine nodes whp.*

PROOF. A cluster C with a fraction p of Byzantine nodes has a probability at most $p(1-\tau)$ to have this fraction decreased by $1/|C|$, and at least $(1-p)\tau$ to have it increased by the same amount. If this fraction is at most $\tau(1+\epsilon/2)$, we prove that it increases by ϵ with probability $o(1/N^\gamma)$, for γ being arbitrarily large depending on the chosen value of k.

The fraction of Byzantine nodes in the cluster is dominated by the martingale with starting state $\tau(1+\epsilon/2)$, which increases or decreases by $1/|C|$ with probability τ. We now show that whp, this martingale will not exceed $\tau(1+\epsilon)$ after $O(\log N)$ steps (recall that $k \log N/l \leq |C| \leq kl \log N$).

For k large enough, let $T^{exchange}$ stands for the number of exchanges. It is $O(\log N)$ and hence there is a constant M such that $T \leq M \log N$. We can derived from Azuma-Hoeffding's inequality that:

$$
\begin{aligned}
Prob(p^C > \tau(1+\epsilon/2)) &< e^{-\epsilon^2/4 \sum_{i=1}^{T^{exchange}} 1/|C|^2} \\
&\leq e^{-\epsilon(k/l)^2 \log^2 N/4(M \log N)} \\
&= e^{-\epsilon(k/l)^2 \log(N)/4M} = n^{-\gamma}
\end{aligned}
$$

□

Similarly, if a cluster has more than a $\tau(1+\epsilon/2)$ fraction of Byzantine nodes, we have that after $O(\log N)$ exchanges, the cluster has less than a $\tau(1+\epsilon/2)$ fraction of Byzantine nodes.

LEMMA 3. *Given a cluster C whose fraction of Byzantine nodes is between $\tau(1+\epsilon)$ and $\tau(1+\epsilon/2)$ (for some constant $\epsilon > 0$ independent of n), then whp, the fraction of Byzantine nodes in this cluster is less than $\tau(1+\epsilon/2)$ after $O(\log N)$ exchanges with nodes chosen uniformly at random.*

PROOF. We use the same arguments for the previous theorem. Here, the fraction of Byzantine node will decrease of $1/|C|$ with probability at least $\tau(1+\epsilon/2)$ and will increase by $1/|C|$ with probability τ. Therefore, as we start from a fraction of at most $\tau(1+\epsilon)$, whp, after $O(\log N)$ exchanges, the fraction of Byzantine nodes in this cluster is less than $\tau(1+\epsilon/2)$. □

When we look at a sequence of s exchanges affecting a given cluster C, we can split this sequence in alternating sub-sequences to apply Lemmas 2 and 3. Some sequences might lead to a fraction of Byzantine nodes between $\tau(1+\epsilon/2)$ and $\tau(1+\epsilon)$, while the following one will lead to a fraction of Byzantine nodes bellow $\tau(1+\epsilon/2)$ whp. Hence, for a sequence s whose length is polynomial, by the union bound, we obtain that is there is always (whp) more than two thirds of honest node in each cluster for an adequate k.

THEOREM 3. *Whp, after a number of steps polynomial in N, at each time step, all clusters are composed of more than two thirds of honest nodes.*

PROOF. Notice that to apply the previous lemmas, one has to ensure that the exchanged nodes are replaced by nodes chosen uniformly at random. This is ensured by our join and leave operations. This is clear for a join operation by the use of a biased CTRW to select the replacement node. For a leave operation, this is also clear for the cluster C from which the node leaves has its nodes exchanged with nodes selected uniformly at random. However, if we look at a cluster C' with which C has exchanged nodes, then the probability that C' receives a Byzantine node is not necessarily τ as it is equal the proportion of Byzantine nodes in C. This is why we enforce C' to exchange all its nodes.

Now, given a specific cluster, C we consider an alternating sequence of time steps $t_1, \ldots, t_i, \ldots$ when the fraction of nodes controlled by the adversary in C becomes larger or equal to $\tau(1+\epsilon/2)$ and when it becomes smaller.

Consider i such that at t_i the fraction of nodes controlled by the adversary in C is less than $\tau(1+\epsilon/2)$ (this is in particular true at the beginning). Then at t_{i+1}, it becomes greater or equal to $\tau(1+\epsilon/2)$ and is less than $\tau(1+\epsilon)$. Lemma 3 ensures that time step t_{i+2} comes within $O(\log N)$ steps, and Lemma 2 ensures that between t_{i+1} and t_{i+2}, the adversary never controls more than a $\tau(1+\epsilon)$ fraction of nodes of the cluster.

By an union bound over all clusters, we have the announced result. $\square$

REMARK 2. *Considering an adversary controlling at most a fraction $1/r - \epsilon$ of the nodes for some constant $\epsilon > 0$ and $r \geq 2$ independent of n, it is possible to strengthen Theorem 3 to obtain that in all the clusters the adversary controls at most a fraction $1/r$ of the nodes.*

5. RELATED WORK

Several authors studied the impact of dynamics on distributed computations [9, 8]. In [10, 21, 22], the communication links of a dynamic network may be modified by the adversary under some connectivity restrictions. In [4], the authors study the scenario in which the adversary can force a large number of nodes of its choice to leave the network while other nodes naturally join the network at the same time. These join and leave operations impact the topology. Yet the size of the network is assumed to remain constant. The authors assume furthermore that the nodes are connected via an expander graph. Depending on whether the adversary has to decide in advance the identities of the nodes to be kicked-out of the network, the authors propose almost-everywhere agreement protocols tolerating at each time step a churn of, respectively $O(n)$ and $O(\sqrt{n})$. The two main differences with our work are that (1) all nodes are assumed to be honest (i.e., the adversary is only external) and (2) nodes are connected via an expander graph by assumption. In contrast, our protocol tolerates a Byzantine adversary controlling a constant fraction of the nodes of the network and dynamically maintains the expander graph.

Some protocols have been proposed to maintain P2P overlay networks. Some offer efficient routing strategies and tolerate crashes, e.g. CAN, Pastry or Tapestry [29, 30, 33]. Some are dedicated to asynchronous networks with concurrent joins and leaves [27]. However, none guarantees both that each node has a low degree and that the resulting overlay exhibits good expansion properties in the sense we require here. Protocols such as SHELL [32] organize peers into a heap structure resilient to large Sybil attacks, while the overlay presented in [23] is resilient to an adversary that can force several peers to crash and join in a arbitrary manner. In [23], the number of join and leave operations tolerated at each turn is proportional to the degree of the nodes, which is optimal. However, the communication cost for maintaining the overlay is high as all the nodes of the network exchange messages at each step.

Other protocols considered unstructured overlays. The protocol of [26] builds an overlay corresponding to an expander graph obtained from the union of several random cycles. This protocol has been further extended and analyzed in [3, 15]. Maintaining unstructured overlays induces fewer message exchanges compared to structured overlays [23, 29, 30, 33] since only a polylogarithmic number of nodes are involved in the communication upon a join or a leave operation. Some of the previous constructions [3, 15] and [26] can be complemented by a recent protocol from Pandurangan and Trehan [28] which preserves the expansion properties of a graph upon adversarial node removals. Nevertheless, the healing procedure proposed does not ensure an absolute expansion factor as we do.

The closest to ours, from the model perspective (dynamic network), is the one developed by Awerbuch and Scheideler [5, 6, 7, 31]. They consider a synchronous network in which an adversary can force nodes to join and leave at each time step, with the constraint that the number of nodes in the network is always within a constant factor of the initial size. Their protocols further require that initially the network is exclusively composed of honest nodes and that the Byzantine ones join the network only after a particular initialization phase has taken place. Within this model, the authors propose a technique to maintain clusters of size $O(\log n)$ composed of a majority of honest ones. Our approach improves upon these previous works in several ways as we do not assume that initially the network is exclusively composed of honest nodes, we describe more precisely how to distributively perform all the operations, and, more importantly, we maintain a partition of the nodes when the size of the network varies polynomially.

6. CONCLUDING REMARKS

This paper answers positively the following question raised in [19]: "Can we [..] address problems of robustness in networks subject to churn? An idea is to assume that: 1) the number of processors fluctuates between n and $\sqrt{n}$ where n is the size of name space; 2) the processors do not know explicitly who is in the system at any time; and 3) that the number of bad processors in the system is always less than a 1/3 fraction. In such a model, can we 1) do Byzantine agreement; and 2) maintain small (i.e. polylogorathimic size) quorums of mostly good processors?"

Our clustering protocol can be leveraged to implement efficient and robust algorithms for various problems such as broadcast, agreement, aggregation, and sampling in the context of highly dynamic networks. A broadcast algorithm using our technique would have for instance $\tilde{O}(n)$ message complexity as compared to $O(n^2)$ without the clustering. Similarly, a sampling algorithm relying on our protocol would have a polylog(n) message complexity per sample.

We currently seek schemes to alleviate the need of the assumption of synchronous nodes. Another objective is to devise a procedure for the initialization phase of NOW whose communication cost is $o(n_{t_0}^2)$ (as opposed to $O(n_{t_0}^3)$).

7. REFERENCES

[1] D. Aldous and J. Fill. Reversible markov chains and random walks on graphs. http://stat-www.berkeley.edu/users/aldous/RWG/book.html.

[2] J. Aspnes and U. Wieder. The expansion and mixing time of skip graphs with applications. *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA'05)*, pages 126–134, 2005.

[3] J. Aspnes and Y. Yin. Distributed algorithms for maintaining dynamic expander graphs. Citeseer, 2008.

[4] J. Augustine, G. Pandurangan, P. Robinson, and E. Upfal. Towards robust and efficient computation in dynamic peer-to-peer networks. *Arxiv preprint*, arXiv:1108.0809, 2011.

[5] B. Awerbuch and C. Scheideler. Group spreading: A protocol for provably secure distributed name service. *Automata, Languages and Programming*, pages 187–210, 2004.

[6] B. Awerbuch and C. Scheideler. Towards scalable and robust overlay networks. In *Proceedings of the International Workshop on Peer-To-Peer Systems (IPTPS'07)*, 2007.

[7] B. Awerbuch and C. Scheideler. Towards a scalable and robust DHT. *Theory of Computing Systems*, 45(2):234–260, 2009.

[8] R. Baldoni, S. Bonomi, and A. S. Nezhad. An algorithm for implementing bft registers in distributed systems with bounded churn. In *SSS*, pages 32–46, 2011.

[9] R. Baldoni, S. Bonomi, and M. Raynal. An implementation in a churn prone environment. In *SIROCCO*, pages 15–29, 2009.

[10] H. Baumann, P. Crescenzi, and P. Fraigniaud. Parsimonious flooding in dynamic graphs. In *Proceedings of the 28th symposium on Principles of distributed computing (PODC'09)*, 2009.

[11] Z. Galil and M. Yung. Partitioned encryption and achieving simultaneity by partitioning. *Information Processing Letters*, 26(2):81 – 88, 1987.

[12] S. Gambs, R. Guerraoui, H. Harkous, F. Huc, and A.-M. Kermarrec. Scalable and secure polling in dynamic distributed networks. *SRDS*, 2012.

[13] J. Garay, J. Katz, R. Kumaresan, and H. Zhou. Adaptively secure broadcast, revisited. In *30th annual Symposium on Principles of Distributed Computing (PODC'11)*, 2011.

[14] A. Giurgiu, R. Guerraoui, K. Huguenin, and A. Kermarrec. Computing in social networks. *Stabilization, Safety, and Security of Distributed Systems*, pages 332–346, 2010.

[15] C. Gkantsidis, M. Mihail, and A. Saberi. Random walks in peer-to-peer networks. In *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'2004)*, volume 1, 2004.

[16] R. Guerraoui, F. Huc, and A.-M. Kermarrec. Highly dynamic distributed computing with byzantine failures. *Arxiv preprint*, arXiv:1202.3084, 2013.

[17] M. Habib, C. McDiarmid, J. Ramirez-Alfonsin, and B. Reed. *Probabilistic Methods for Algorithmic Discrete Mathematics*. Springer Verlag, Berlin, 1998.

[18] V. King, S. Lonargan, J. Saia, and A. Trehan. Load balanced scalable byzantine agreement through quorum building, with full information. *Distributed Computing and Networking*, pages 203–214, 2011.

[19] V. King and J. Saia. Scalable Byzantine Computation. *ACM SIGACT News*, 41(3), 2010.

[20] M. Krebs and A. Shaheen. *Expander Families and Cayley Graphs: A Beginner's Guide*. Oxford University Press, USA, 2011.

[21] F. Kuhn, N. Lynch, and R. Oshman. Distributed computation in dynamic networks. In *Proceedings of the 42nd symposium on Theory of computing (STOC'10)*, pages 513–522, 2010.

[22] F. Kuhn, Y. Moses, and R. Oshman. Coordinated consensus in dynamic networks. In *Proceedings of the 30th annual symposium on Principles of distributed computing (PODC'11)*, pages 1–10, 2011.

[23] F. Kuhn, S. Schmid, and R. Wattenhofer. Towards worst-case churn resistant peer-to-peer systems. *Distributed Computing*, 22(4):249–267, 2010.

[24] K. L., E. Liberty, and O. Somekh. Estimating sizes of social networks via biased sampling. In *Proceedings of the 20th international conference on World wide web*, pages 597–606. ACM, 2011.

[25] L. Lamport, R. Shostak, and M. Pease. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems (TOPLAS'82)*, 4(3):382–401, July 1982.

[26] C. Law and K. Siu. Distributed construction of random expander graphs. In *Proceedings of 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'2003)*, pages 2133–2143, 2003.

[27] X. Li, J. Misra, and C. Plaxton. Active and concurrent topology maintenance. *Distributed Computing*, pages 320–334, 2004.

[28] G. Pandurangan and A. Trehan. Xheal: localized self-healing using expanders. In *Proceedings of the 30th annual symposium on Principles of distributed computing (PODC'11)*, pages 301–310, 2011.

[29] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172, 2001.

[30] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proceedings of Middleware'01*, pages 329–350, 2001.

[31] C. Scheideler. How to spread adversarial nodes? Rotate! *Proceedings of the 37th annual symposium on Theory of computing (STOC'05)*, page 704, 2005.

[32] C. Scheideler and S. Schmid. A distributed and oblivious heap. *Automata, Languages and Programming*, pages 571–582, 2009.

[33] B. Zhao, L. Huang, J. Stribling, S. Rhea, A. Joseph, and J. Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1):41–53, 2004.