Performance evaluation of large-scale dynamic systems
Emmanuelle Anceaume, Romaric Ludinard, Bruno Sericola

To cite this version:

HAL Id: hal-00736918
https://hal.archives-ouvertes.fr/hal-00736918
Submitted on 30 Sep 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
In this paper we present an in-depth study of the dynamicity and robustness properties of large-scale distributed systems, and in particular of peer-to-peer systems. When designing such systems, two major issues need to be faced. First, population of these systems evolve continuously (nodes can join and leave the system as often as they wish without any central authority in charge of their control), and second, these systems being open, one needs to defend against the presence of malicious nodes that try to subvert the system. Given robust operations and adversarial strategies, we propose an analytical model of the local behavior of clusters, based on Markov chains. This local model provides an evaluation of the impact of malicious behaviors on the correctness of the system. Moreover, this local model is used to evaluate analytically the performance of the global system, allowing to characterize the global behavior of the system with respect to its dynamics and to the presence of malicious nodes and then to validate our approach.

1. INTRODUCTION

The adoption of peer-to-peer overlay networks as a building block for architecting Internet scale systems has firstly raised the attention of making these systems resilient to nodes-driven dynamics. This dynamics—that represents the propenseness of thousands or millions of nodes to continuously join and leave the system, and which is usually called churn—if not efficiently managed, quickly gives rise to dropped messages and data inconsistency, and thus to an increasing latency and bandwidth due to repairing mechanisms. The other fundamental issue faced by any practical open system is the inevitable presence of malicious nodes [1, 2, 3]. Guaranteeing the liveness and safety of these systems requires their ability to self-heal or at least to self-protect against this adversity. Malicious nodes can devise complex strategies to subvert the system. In particular these attacks aimed at exhausting key resources of hosts (e.g., bandwidth, CPU processing, TCP connection resources) to diminish their capacity to provide or receive services [4]. Different approaches have been proposed to face adversarial setting, each one focusing on a particular adversary strategy. Regarding eclipse attacks, a very common technique, called constrained routing table, relies on the uniqueness and impossibility of forging nodes identifiers. It consists in selecting neighbors based on their identifiers so that all of them are close to some particular points in the identifier space [1]. Such an approach has been implemented into several overlays (e.g., [5, 6, 7]). To prevent messages from being mis-routed or dropped, the seminal works on routing security by Castro et al. [1] and Sit and Morris [8] combine routing failure tests and redundant routing as a solution to ensure robust routing. Their approach has then been successfully implemented in different structured-based overlays (e.g., [9, 10, 11]). However all these solutions assume that malicious nodes are uniformly distributed in the system. Actually, malicious nodes may concentrate their power to attack some region of the overlay [12, 13]. These targeted attacks require additional mechanisms to be tolerated or at least to be confined. Actually, it has been shown [14] that structured overlays cannot survive targeted attacks if the adversary may keep sufficiently long its malicious peers at the same position in the overlay. Indeed, once malicious peers have succeeded in sitting in a focused region of the overlay, they can progressively gain the quorum within this region by simply waiting for honest peers to leave their position, leading to the progressive isolation of honest peers. The two fundamental properties that prevent peers isolation are the guarantee that the distribution of peers identifiers is random, and that peers cannot stay forever at the same position in the system [12].

In the present work, we present an in-depth study of the dynamic and robustness aspects of large-scale distributed systems. We investigate adversarial strategies that aim at isolating honest nodes in cluster-based overlays. We propose an analytical model of the local behavior of clusters, based on Markov chains. This local model provides an evaluation of the power of malicious behaviors. We use this local model to evaluate analytically the performance of the global system, allowing to characterize the global behavior of the system with respect to its dynamics and to the presence of malicious nodes and then to validate our approach. Our analysis shows that i) by gathering nodes in clusters, ii) by preventing nodes to stay infinitely long at the same position in the overlay and iii) by introducing randomness in the operations of the overlay, the impact of the adversary on the cluster correctness is very limited. In particular we show that the adversary has no incentive to trigger topological operations on the system which confines its impact on particular regions of the system.

The current work extends the study done in [13], where the focus was placed on the local behavior of a cluster rather than on the global behavior of the system as done in the present work. Specifically, [13] studies according to different levels of adversity in the cluster, the impact of randomized operations on the time spent by a cluster in safe and corrupted states, and on the proportion of clusters that split or
merge in a safe or corrupted state. The main lessons drawn
from that study is that a small amount of randomization is
sufficient to defend against malicious nodes, even when
they collude together to increase their power. This result
is interesting because it notably decreases the complexity of
the overlay operations.

The remainder of this paper is as follows: In Section 2, we
briehly present the main features of structured based over-
lays. In Section 3 we present an analytical model of the local
behavior of a cluster based on Markov chains. Section 4 uses
this model to evaluate the performance of the global over-
lay. This analysis is applied to a large scale dynamic system
presented in Section 5. Section 6 concludes.

2. CLUSTER-BASED SYSTEMS

2.1 Self-organization of Nodes

We consider a dynamic system populated by a large col-
lection of nodes in which each node is assigned a unique and
permanent random identifier from an $m$-bit identifier space.
Node identifiers (simply denoted $ids$ in the following) are de-
derived by applying some standard strong cryptographic hash
function on nodes intrinsic characteristics. The value of $m$
(160 for the standard SHA1 hash function) is chosen to be
large enough to make the probability of identifiers collision
negligible. Each application-specific object, or data-item, of
the system is assigned a unique identifier, called key, selected
from the same $m$-bit identifier space. Each node $p$ owns a
fraction of all the data items of the system. The system is
subject to churn, which is classically defined as the rate of
turnover of nodes in the system [15]. For scalability reasons,
each node locally knows only a small set of nodes existing
within the system. This set is typically called the node’s
local view or the node’s neighborhood. The particular algo-
rithm used by nodes to build their local view and to route
messages induces the resulting overlay topology. Structured
overlays (also called Distributed Hash Tables (DHTs)) build
their topology according to structured graphs (e.g., hyper-
cube, torus). Specifically, nodes self-organize within the
structured graph according to a distance function $D$ based
on their ids, plus possibly other criteria such as geographi-
cal distance. Finally, the mapping between nodes and data
derives from the same distance function $D$.

In cluster-based overlays, clusters of nodes substitute for
nodes at the vertices of the structured graph. Each vertex of
the structured graph is composed of a set or cluster of nodes.
Clusters in the overlay are uniquely identified. Cluster size $s$
evolves according to join and leave events. However for
robustness and scalability reasons, $s$ is both lower and upper
bounded. The lower bound $S_{\text{min}}$ satisfies some constraint
based on the assumed failure model, while the upper bound
$S_{\text{max}}$ is typically in $O(\log N)$ where $N$ is the current number
of nodes in the overlay. Nodes join the clusters according to
distance $D$. For instance in PeerCube [10], $p$ joins the
(unique) cluster whose identifier is a prefix of $p$’s identifier.
Once a cluster size exceeds $S_{\text{max}}$, this cluster splits into
two smaller clusters, each one populated with the nodes that
are closer to each other according to distance $D$. Nodes can freely leave their cluster. Once a cluster undershoots its minimal size $S_{\text{min}}$, this cluster merges with the closest cluster in its neighborhood.

In the following we assume that join and leave events have an equal chance to occur in any cluster.

2.2 Malicious Behaviors

We assume the presence of malicious nodes (also called
Byzantine nodes in the distributed computing community)
that try to subvert the whole system by exhibiting unde-
sirable behaviors [8, 16]. In our context this amount to
devising strategies to take the control of clusters so that
all the queries that go through these controlled clusters can
be deviated toward malicious nodes (this refers to eclipse
attacks [1, 8]), and all the data objects stored at these clus-
ters can be manipulated. They can magnify their impact by
colluding and coordinating their behavior. We model these
strategies through a strong adversary that controls these
malicious peers. A strong adversary is an adversary allowed
to deviate arbitrarily far from the protocol specification. We
assume that the adversary has large but bounded resources
in that it cannot control more than a fraction $\mu$ ($0 < \mu < 1$)
of malicious nodes in the whole network. Note that in the
following we make a difference between the whole system
and the overlay. The system $\mathcal{U}$ encompasses all the nodes
that at some point may participate to the overlay (i.e., $\mathcal{U}$
contains up to $2^n$ nodes), while the overlay $\mathcal{X}$ contains at
any time the subset of participating nodes (i.e, $N$ size is
equal to $N \leq 2^n$). Thus, while $\mu$ represents the assumed
fraction of malicious nodes in $\mathcal{U}$, the goal of the adversary
is to populate the overlay with a larger fraction of malicious
nodes in order to subvert its correct functioning. Finally, a
node which always follows the prescribed protocols is said
to be honest. Note that honest nodes cannot $a$\textit{ priori} dis-
tinguish honest nodes from malicious ones.

2.3 Operations of the Overlay

From the application point of view, two key operations are
provided by the system: the join operation that enables a
node to join the closest cluster in the system to itself accord-
ing to distance $D$, and the leave operation, indicating that
some node leaves its cluster. Note that other operations are
also provided to the application (e.g., the lookup($k$) opera-
tion that enables a node to search for key $k$, and the put($x$
operation, which allows it to insert data $x$ in the system),
however they have no impact on the dynamic of the system
and a minor one on its robustness. Thus we only concentrate
on analyzing and evaluating the dynamics and robustness of
the system through the join and leave operations. Now,
from the topology structure point of view, two operations
can result in a topology modification and thus must be
taken into account to evaluate the dynamics of the system,
namely the split and the merge operations. When the size of
a cluster exceeds $S_{\text{max}}$, this cluster splits into two new
clusters, and when the size of a cluster goes below $S_{\text{min}}$, this
cluster merges with other clusters to guarantee the cluster resil-
liency. Note that for robustness reasons, a cluster may
have to temporarily exceed its maximal size $S_{\text{max}}$ before be-
ing able to split into two new clusters. This guarantees that
resiliency of both new clusters is met, i.e., both clusters
sizes are at least equal to $S_{\text{min}}$. For the time being, we do
not need to go further into the details of these operations,
however Section 5 will precise them.

3. MODELING THE CLUSTER BEHAVIOUR

We model the effect of join and leave events in a cluster
using a homogeneous discrete-time Markov chain denoted by
$X = \{X_n, n \geq 0\}$. Markov chain $X$ represents the
evolution of the number of honest and malicious nodes in
the cluster. The impact of malicious nodes on the cluster correctness is application dependent. For instance a necessary and sufficient condition to prevent agreement among a set of nodes is that strictly more than a third of the population set is malicious [17]. In this Section, the conditions under which cluster correctness holds are lumped in predicate correct. This predicate will be explicit in Section 5 where the application is described. Thus we define the state of a cluster as safe if predicate correct holds, while it is defined as attacked otherwise. Markov chain \( X \) alternates between the set of all the secure states, denoted by \( S \), and the set of all the attacked states, denoted by \( A \), until entering the absorbing state (cf. Figure 1).

This state, denoted by \( A \), represents the logical disappearance of a cluster from the graph. This occurs when the cluster either splits into two smaller clusters (i.e., it has reached its maximal size \( S_{\text{max}} \)) or merges with its closest neighbor (i.e., it has reached its minimal size \( S_{\text{min}} \)).

The transition matrix of \( X \), denoted by \( P \), is partitioned with respect to the decomposition of the state space \( \Omega = S \cup A \cup a \)

\[
P = \begin{pmatrix}
P_S & P_{SA} & P_{Sa} \\
P_{AS} & P_A & P_{Aa} \\
0 & 1 & 0
\end{pmatrix}
\]

where \( P_{UV} \) is the sub-matrix of dimension \(|U| \times |V|\) containing the transitions from states of \( U \) to states of \( V \), with \( U, V \in \{S, A, a\} \). We simply write \( P_U \) for \( P_{UU} \).

In the same way, the initial probability distribution \( \alpha \) is partitioned by writing \( \alpha = (\alpha_S \, \alpha_A \, \alpha_a) \), where \( \alpha_a = 0 \) and the sub-vector \( \alpha_U \) contains the initial probabilities of states \( U \in \{S, A\} \).

4. MODELING THE OVERLAY NETWORK

4.1 Notations

We consider an overlay populated with \( n \) clusters denoted by \( D_1, \ldots, D_n \) and subject to join and leave events. Each cluster \( D_i \) implements the same join, leave, split and merge operations. We assume that join and leave events are uniformly distributed throughout the overlay. Specifically, when a join or leave event occurs in the overlay it affects cluster \( D_i \) with probability \( p_i = 1/n \). Thus we consider, for \( n \geq 1 \), \( n \) Markov chains \( X^{(1)}, \ldots, X^{(n)} \) identical to \( X \), i.e., with the same state space \( \Omega \), the same transition probability matrix \( P \) and the same initial probability distribution \( \alpha \). However these chains are not independent since, at each instant, only one Markov chain, chosen with probability \( 1/n \), is allowed to make a transition.

We denote by \( N_S^{(m)}(m) \) and \( N_A^{(m)}(m) \) the respective number of safe and polluted clusters just after the \( m \)-th join or leave event, i.e., the respective number of Markov chains that are in set \( S \) and in set \( A \) at instant \( m \). More formally, these random variables are defined, for \( m \geq 0 \), by

\[
N_S^{(m)}(m) = \sum_{h=1}^{m} \mathbf{1}_{\{X_h^{(n)} \in S\}} \quad \text{and} \quad N_A^{(m)}(m) = \sum_{h=1}^{m} \mathbf{1}_{\{X_h^{(n)} \in A\}}
\]

where notation \( \mathbf{1}_{\{A\}} \) represents the indicator function, which is equal to 1 if condition \( A \) is true and 0 otherwise. We denote by \( \mathbb{1}_S \) (resp. \( \mathbb{1}_A \)) the column vector of dimension \(|S \cup A|\) whose \( i \)-th entry is equal to 1 (resp. 0) if \( i \in S \) (resp. 1) if \( i \in A \). We denote by \( \mathbb{1}_P \) the sub-matrix of \( P \) containing the transitions between the states of \( S \cup A \), i.e., the matrix obtained from \( P \) by removing the row and the column corresponding to the absorbing state \( a \). The dimension of \( T \) is thus \(|S \cup A|\).

Finally, we define \( \alpha_T = (\alpha_S \, \alpha_A) \). In [13], we studied the expectations of both random variables \( N_S^{(m)}(m) \) and \( N_A^{(m)}(m) \). In what follows we focus on their joint distribution.

4.2 Focus on the First Topological Change

We study the evolution of the \( n \) Markov chains in the set of safe states and in the set of polluted states at time \( m \) respectively when none of the \( n \) Markov chains are absorbed at time \( m \).

For \( v = 0, \ldots, n \), we are interested in the probability \( p_v^{(n)}(m) \) defined by

\[
p_v^{(n)}(m) = \mathbb{P} \{ N_S^{(m)}(m) = n - v, N_A^{(m)}(m) = v \},
\]

which represents the probability that \( v \) Markov chains are in polluted states at time \( m \) and none of the \( n \) Markov chains are absorbed at time \( m \).

For every \( m \geq 0 \) and \( \ell \geq 1 \), we define the set \( S_{m,\ell} \) as

\[
S_{m,\ell} = \{m = (m_1, \ldots, m_{\ell}) \in \mathbb{N}^\ell \mid m_1 + \cdots + m_\ell = m\}.
\]

The probability \( p_v^{(n)}(m) \) is given by the following theorem where, as usual, we take as convention that an empty product is equal to 1.

We introduce the notation

\[
q_S(k) = \mathbb{P} \{ X_k \in S \} = \alpha_T T^k \mathbb{1}_S
\]

and

\[
q_A(k) = \mathbb{P} \{ X_k \in A \} = \alpha_T T^k \mathbb{1}_A.
\]

Theorem 1. For every \( m \geq 0 \) and \( v = 0, \ldots, n \), we have

\[
p_v^{(n)}(m) = \binom{n}{v} \alpha_T T^m \sum_{w \in S_{m,n}} \frac{m!}{m_1! \cdots m_n!} \prod_{r=1}^{v} q_A(m_r) \prod_{r=v+1}^{n} q_S(m_r).
\]

Proof. We introduce the set \( H = \{1, \ldots, n\} \) and, for \( v = 0, \ldots, n \), the set \( H(v) \) of all subsets of \( H \) with \( v \) elements, i.e.,

\[
H(v) = \{ C \subseteq H \mid |C| = v \}.
\]

It is easily checked that \(|H(v)| = \binom{n}{v}\). With this notation, we have

\[
p_v^{(n)}(m) = \sum_{C \in H(v)} \mathbb{P} \{ X_m^{(n)} \in A \cap C \} = \sum_{C \in H(v)} \mathbb{P} \{ X_m^{(n)} \in S \} = \sum_{C \in H(v)} \mathbb{P} \{ X_m^{(n)} \in C \}.
\]

The probability mass distribution, used to choose at each instant the Markov chain that has to do a transition, being uniform, all the probabilities in the above sum are equal.

We thus have, by taking \( C = \{1, \ldots, v\} \) and using Theorem

Figure 1: Aggregated view of Markov chain \( X \).
The random variable $\Theta$ is defined by

$$
P_{\nu}^{(n)}(m) = \left(\frac{n}{v}\right) \mathbb{P}\{X_m^{(v)} \in A, \ldots, X_m^{(v)} \in A, X_m^{(v+1)} \in S, \ldots, X_m^{(n)} \in S\} = \left(\frac{n}{v}\right) \frac{1}{n^m} \sum_{m \in S_m} \frac{m!}{m_1! \cdots m_n!} \prod_{r=1}^n q_A(m_r) \prod_{r=v+1}^n q_S(m_r),
$$

which completes the proof. □

Theorem 2 gives a recurrence relation to compute the probabilities $p_{\nu}^{(n)}(m)$ with a polynomial complexity.

**Theorem 2.** For every $v = 0, \ldots, n - 1$, we have

$$
p_{\nu}^{(n)}(m) = \left(\frac{n}{v}\right) \frac{1}{n^m} \sum_{m \in S_{m-n}} \frac{m!}{m_1! \cdots m_{m-n-1}!} \prod_{r=1}^n q_A(m_r) \prod_{r=v+1}^n q_S(m_r)
$$

and, for $v = n$,

$$
p_{\nu}^{(n)}(m) = \left(\frac{n}{v}\right) \frac{1}{n^m} \sum_{m \in S_{m-n-1}} \frac{m!}{m_1! \cdots m_{m-n-1}!} \prod_{r=1}^n q_A(m_r) \prod_{r=v+1}^n q_S(m_r)
$$

PROOF. Extracting in the sum (1) the index $m_n$, renaming it $\ell$ and using Theorem 1 for integers $n - 1$ and $m - \ell$, we get, for $v = 0, \ldots, n - 1$,

$$
p_{\nu}^{(n)}(m) = \left(\frac{n}{v}\right) \frac{1}{n^m} \sum_{m \in S_{m-n}} \frac{m!}{m_1! \cdots m_{m-n-1}!} \prod_{r=1}^n q_A(m_r) \prod_{r=v+1}^n q_S(m_r)
$$

The second part of the proof is obtained similarly. □

Let $\Theta_n$ be the first instant at which one of the $n$ Markov chains $X^{(1)}, \ldots, X^{(n)}$ gets absorbed. This random variable is defined by

$$
\Theta_n = \inf\{m \geq 0 \mid \exists r \text{ such that } X_m^{(r)} = a\}.
$$

The random variable $\Theta_n$ has been studied in [18]. In particular, we have shown that its distribution is given for every $k \geq 0$ and $u = 1, \ldots, n - 1$ by

$$
P\{\Theta_n > k\} = \sum_{\ell=0}^k \binom{k}{\ell} \left(\frac{n}{v}\right)^\ell \left(1 - \frac{n}{v}\right)^{k-\ell} \times P\{\Theta_u > \ell\} P\{\Theta_{n-u} > k-\ell\}.
$$

By taking $n = 2^m$ and $u = 2^{m-1}$, we have

$$
P\{\Theta_{2^m} > k\} = \frac{1}{2^m} \sum_{\ell=0}^k \binom{k}{\ell} P\{\Theta_{2^{m-1}} > \ell\} P\{\Theta_{2^{m-1}} > k-\ell\}.
$$

Relation (2) is particularly interesting for very large values of $n$. Indeed, the complexity for the computation of the distribution and the expectation of $\Theta_n$ is in $O(\log_2 n)$. Note also that this relation can be split into 2 identical sums plus the central term corresponding to $\ell = k/2$ when $k$ is even.

We study here the expected number of Markov chains which are in a safe state and in an attacked state at time $m$ respectively when none of the $n$ Markov chains are absorbed at time $m$.

The quantity $E(N_S^{(n)}(m)1_{(\Theta_n > m)})$ is the expected number of Markov chains that are in a safe state at time $m$, for $\Theta_n > m$. In the same way, we define the expected value $E(N_A^{(n)}(m)1_{(\Theta_n > m)})$ for attacked states.

Remark that, for every $v = 0, \ldots, n$, we have

$$
(N_S^{(n)}(m) = v, N_A^{(n)}(m) = n-v) \Leftrightarrow (N_S^{(n)}(m) = v, \Theta_n > m).
$$

Using this remark, we obtain the following result.

**Theorem 3.** For every $m \geq 0$, we have

$$
E(N_S^{(n)}(m)1_{(\Theta_n > m)}) = \sum_{v=1}^n vP\{N_S^{(n)}(m) = v, \Theta_n > m\} = \sum_{v=1}^n vP_{\nu}^{(n)}(m) = \sum_{v=1}^n \frac{n}{v} \frac{1}{n^m} \sum_{m \in S_{m-n}} \frac{m!}{m_1! \cdots m_{m-n-1}!} \prod_{r=1}^n q_A(m_r) \prod_{r=v+1}^n q_S(m_r)
$$

PROOF. From the previous remark, we get

$$
E(N_S^{(n)}(m)1_{(\Theta_n > m)}) = \sum_{v=1}^n \frac{n}{v} \frac{1}{n^m} \sum_{m \in S_{m-n}} \frac{m!}{m_1! \cdots m_{m-n-1}!} \prod_{r=1}^n q_A(m_r) \prod_{r=v+1}^n q_S(m_r)
$$

Using the first relation of Theorem 2, this leads to

$$
E(N_S^{(n)}(m)1_{(\Theta_n > m)}) = \sum_{v=1}^n \frac{n}{v} \frac{1}{n^m} \sum_{m \in S_{m-n}} \frac{m!}{m_1! \cdots m_{m-n-1}!} \prod_{r=1}^n q_A(m_r) \prod_{r=v+1}^n q_S(m_r)
$$

Since

$$
\sum_{v=1}^n \frac{n}{v} \frac{1}{n^m} \sum_{m \in S_{m-n}} \frac{m!}{m_1! \cdots m_{m-n-1}!} \prod_{r=1}^n q_A(m_r) \prod_{r=v+1}^n q_S(m_r)
$$

we get the desired result. Relation (4) is obtained using the same lines. □

It has been shown in [18] that for all $k \geq 0$

$$
\lim_{n \to +\infty} P\{\Theta_n > k\} = (\alpha T \mathbb{P})^k
$$

4
where \( \mathbf{1} \) is the column vector containing only ones with appropriate dimension determined by the context. Using this result and Relations (3) and (4) we get

\[
\lim_{n \to \infty} \frac{E(N_S(\alpha)(m)1_{1_{(\sigma_n > m)}})}{n} = q_S(0)(\alpha T T_1)^m = \alpha S \mathbf{1} (\alpha T T_1)^m
\]

\[
\lim_{n \to \infty} \frac{E(N_A(\alpha)(m)1_{1_{(\sigma_n > m)}})}{n} = q_A(0)(\alpha T T_1)^m = \alpha A \mathbf{1} (\alpha T T_1)^m.
\]

5. APPLICATION

Internet growth in recent years has motivated an increasing research in peer-to-peer systems to implement different applications such as file sharing, large scale distribution of data, streaming, and video-on-demand. All these applications share in common large data and the necessity of durable access to them. Implementing these applications over a peer-to-peer system allows to potentially benefit from the very large storage space globally provided by the many unused or idle machines connected to the network. This has led for the last few years to the deployment of a rich number of large scale storage architectures. Typically, reliable storage amounts to replicating data sufficiently enough so that despite adversarial behaviors replicas are still reachable. Given the specificities of the P2P paradigm, the replication scheme must guarantee a low message overhead (one cannot afford to store many copies of very large original data), and low bandwidth requirements (upon unpredictable join and leave of nodes it is eventually necessary to recreate new copies of the data to keep a certain number of replicas). Rateless erasure coding [19, 20, 21] (also called Fountain coding) meets these constraints. Specifically, an object is divided into \( k \) equal size fragments, and recoded into a potentially unlimited number of independent check blocks. Fundamental property of erasure coding is that one may recover an initial object by collecting \( k' \) distinct check blocks generated by different sources, with \( k' = k(1 + \epsilon) \) with \( \epsilon \) arbitrarily small.

We propose to take advantage of cluster-based overlays to store these check blocks. Specifically, each of these pieces of data is assigned a unique identifier, called key, selected from the same \( m \)-bit identifier space as nodes one (cf. Section 2). Each piece of data \( o \) is then disseminated to the cluster \( D_i \) whose label is closer to \( o \) key. To take advantage of the natural redundancy present in each cluster while efficiently handling the churn of the system, the population of each cluster is organized into two sets: the core set and the spare set. Members of the core set are primarily responsible for handling messages routing, and clusters operations. Management of the core set is such that its size is maintained to constant \( S_{\min} \). This constant is defined according to the assumed proportion of malicious nodes in the system. Spare members are the complement number of nodes in the cluster. Size \( s \) of the spare set is such that \( 0 \leq s \leq S_{\max} - S_{\min} \). For scalability reasons, \( S_{\max} \) is in \( O(\log N) \) where \( N \) is the current number of nodes in the overlay. In contrast to core members, spare members are not involved in any of the overlay operations, however they are in charge of data storage. To handle malicious behaviors, robust join, leave, split and merge operations are designed as detailed in Section 5.1. In addition to robust operations, we propose to limit the lifetime of node identifiers and to randomize their computation. This is achieved by adding an incarnation number \( t_o \) to the fields that appear in the node certificate (certificates are acquired at trustworthy Certification Authorities (CAs)), and by hashing this certificate to generate the initial node identifier \( id^0 \). The incarnation number limits the lifetime of identifiers and thus the position of the nodes in the overlay. The current incarnation \( k \) of any node \( p \) can be computed as \( k = \lfloor (t - t_0)/t \rfloor \), where \( t \) is the current time, and \( t \) is the length of the lifetime of each node incarnation. Thus, the \( k^{th} \) incarnation of node \( p \) expires when its local clock reads \( t_0 + k \ell \). At this time \( p \) must rejoin the system using its \((k + 1)^{th}\) incarnation. By the properties of hash functions, this guarantees that nodes are regularly pushed toward unpredictable and random positions of the overlay. At any time, any node can check the validity of the identifier of any other node \( q \) in the system, by simply calculating the current incarnation \( k \) of \( q \) and generating the corresponding identifier. This leads to the following property.

Property 1 (Limited Sojourn Time). Let \( D_i \) be some cluster of the system and \( p \) some peer in the overlay. Then \( q \) belongs to \( D_i \) at time \( t \) if and only if \( id^t \), matches the label of \( D_i \) according to distance \( D \) (we say that \( q \) is valid for \( D_i \)).

5.1 Robust Operations

To protect the system against the presence of malicious nodes in the overlay, we propose to take advantage of nodes role separation at cluster level to design robust operations. Briefly, the join operation is designed so that brute force denial of service attacks are discouraged. The Leave operation impedes the adversary from predicting what is going to be the composition of the core set after a given sequence of join and leave events triggered by its malicious nodes. Finally, as both merge and split operations induce topological changes in the overlay, and more importantly may have an influence on the subset of the identifier space the adversary may gain control over\(^1\), these operations have been designed so that the adversary has, in expectation, no interest to trigger them. Specifically, these four operations make up the overlay protocol and are specified as follows

- **join(p):** when a node \( p \) joins the system, it joins the spare set of the closest cluster in the system (according to distance \( D \)). Core members of this cluster update their spare view to reflect \( p \)’s insertion (note that the spare view update does not need to be tightly synchronized at all core members).

- **leave(p):** When a node \( p \) leaves a cluster either \( p \) belongs to the spare set or to the core set. In the former case, core members simply update their spare view to reflect \( p \)’s departure, while in the latter case, the core view maintenance procedure is triggered. This procedure consists in replacing \( p \) with one node randomly chosen from the spare set.

- **split(\( D_i \)):** when a cluster \( D_i \) has reached the conditions to split into two smaller clusters \( D_{i1} \) and \( D_{i2} \), core sets of both \( D_{i1} \) and \( D_{i2} \) are built. Priority is given

\(^1\)Indeed, a merge operation doubles the subset of the identifier space a cluster is responsible for, while a split operation divides it per two.
$$0 < s < \Delta$$

<table>
<thead>
<tr>
<th>Probabilities</th>
<th>Value</th>
<th>Meaning of the probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu$</td>
<td>ratio of Byzantine peers in the universe $U$</td>
<td></td>
</tr>
<tr>
<td>$S_{\text{max}}$</td>
<td>maximal size of a cluster</td>
<td></td>
</tr>
<tr>
<td>$S_{\text{min}}$</td>
<td>size of the core set of a cluster ($S_{\text{min}}$ is a system parameter)</td>
<td></td>
</tr>
<tr>
<td>$\Delta$</td>
<td>$S_{\text{max}} - S_{\text{min}}$</td>
<td></td>
</tr>
<tr>
<td>$s$</td>
<td>current size of the spare set</td>
<td></td>
</tr>
<tr>
<td>$x$</td>
<td>number of malicious peers in the spare set</td>
<td></td>
</tr>
<tr>
<td>$y$</td>
<td>number of malicious peers in the core set</td>
<td></td>
</tr>
<tr>
<td>$d$</td>
<td>probability that the lifetime of a given peer identifier has not expired (per unit of time)</td>
<td></td>
</tr>
<tr>
<td>$p_j$ (resp. $p_l$)</td>
<td>$1/2$</td>
<td>join (resp. leave) event probability</td>
</tr>
<tr>
<td>$p_c$</td>
<td>$S_{\text{min}}/(S_{\text{min}} + s)$</td>
<td>probability for a peer to belong to the cluster core set</td>
</tr>
<tr>
<td>$p_m$</td>
<td>$\mu$</td>
<td>probability that the joined peer is malicious</td>
</tr>
<tr>
<td>$p_{mc}$</td>
<td>$x/S_{\text{min}}$</td>
<td>probability for a core member to be malicious</td>
</tr>
<tr>
<td>$p_{ms}$</td>
<td>$y/s$</td>
<td>probability for a spare member to be malicious</td>
</tr>
<tr>
<td>$1 - d^+$</td>
<td>probability that Property 1 is satisfied in the core set during one unit of time</td>
<td></td>
</tr>
<tr>
<td>$1 - d^+$</td>
<td>probability that Property 1 is satisfied in the spare set during one unit of time</td>
<td></td>
</tr>
<tr>
<td>$1(\Delta)$</td>
<td>1 if condition $A$ is true, 0 otherwise</td>
<td>represents the indicator function</td>
</tr>
</tbody>
</table>

Figure 2: Transition diagram for the computation of the transition probability matrix $P$.  

6
to core members of \(D_t\) and completion is done with randomly chosen spares in \(D_t\). This random choice is handled through a Byzantine-tolerant consensus run among core members of \(D_t\). Spares members of \(D_t\) (resp. \(D_s\)) are populated with the remaining spares members of \(D_t\) that are closer to \(D_s\) than to \(D_t\) (resp. closer to \(D_t\) than to \(D_s\)).

- **merge**\((D_s, D_t)\): when some cluster \(D_s\) has reached the conditions to merge (i.e., its spare set is empty), it merges with the closest cluster \(D_t\) to \(D_s\). The created cluster \(D_t\) is composed by a core set whose members are the core set members of \(D_s\) and by a spare set whose members are the union of the spare members of \(D_s\) and the core set members of \(D_t\).

### 5.2 Specification of the Adversarial Behavior

Based on the operations described here above, we investigate how malicious nodes could proceed to compromise clusters correctness. Correctness of any cluster \(D_t\) is jeopardized as soon as \(D_t\) core set is populated by more than a quorum \(c\) of malicious nodes where \(c = \lfloor (S_{\text{min}} - 1)/3\rfloor\).

As a consequence of assigning an initial unique random node id to nodes and of periodically pushing them to random regions in the overlay, the strategy of the adversary to increase the global representation of malicious identifiers is a combination of the following two actions

- maximizing the number of malicious nodes that sit in the whole overlay, and
- minimizing the likelihood that any attacked cluster \(D_t\) switches back to a safe state.

Note that there is a trade-off between these two strategies. Indeed, while in the former case, the adversary aims at triggering as many \textit{join} operations as possible for its malicious nodes to be present into the clusters, in the latter case, the adversary has no interest to let clusters grow in such a way that these clusters will undergo a \textit{split} operation (cf. Section 5.1). Indeed, the outcome of a \textit{split} operation cannot increase the subset of identifiers space the adversary has gained control over—at best, it keeps it the same. Thus when an attacked cluster \(D_t\) is close to \textit{split}, the adversary does not trigger anymore malicious \textit{join} operations that would lead malicious nodes to join \(D_t\), and prevents honest nodes from joining \(D_t\) whenever the size \(s\) of the spare set satisfies \(s > 1\). This guarantees that \(D_t\) will not grow because of honest peers, while minimizing the likelihood that \(D_t\) undergoes a \textit{merge} operation as well. The reason is that by construction of the \textit{merge} operation (cf. Section 5.1), when \(D_t\) triggers a \textit{merge} operation with its closest neighbor then all the members of \(D_t\) are pushed to the spare set of the new created cluster which clearly deters the adversary from triggering \textit{merge} operations. This leads to the following rule

#### Rule 1 (Adversarial Join Strategy)

Let \(D_t\) be a cluster such that at time \(t\) its core set contains \(\ell > c\) valid malicious nodes. Any join event issued by node \(q\) and received at \(D_t\) at time \(t\) is discarded if \((q\) is honest and \(s > 1\)) or \((s = S_{\text{max}} - S_{\text{min}} - 1)\).

To summarize, the strategy of the adversary is to maximize the whole subset of the identifiers space it has gained control over. This is achieved by first never asking its malicious peers to leave their cluster except if Property 1 does not hold, and second by having the maximal number of malicious peers join the system except if Rule 1 holds.

### 5.3 Modeling the Evolution of the Cluster

We can now instantiate Markov chain \(X = \{X_n, n \geq 0\}\) (introduced in Section 3), with the specificities of our application context. Specifically, for \(n \geq 1\), the event \(X_n = (s, x, y)\) represents the state of a cluster after the \(n\)-th transition (i.e., the \(n\)-th join or leave event), so that the size of the spare set is equal to \(s\), the number of malicious peers in the core set is equal to \(x\) and the number of malicious peers in the spare set is equal to \(y\). The state space \(\Omega\) of \(X\) is defined by \(\Omega = \{(s, x, y) | 0 \leq s \leq \Delta, 0 \leq x \leq S_{\text{min}}, 0 \leq y \leq s\}\), where \(\Delta = S_{\text{max}} - S_{\text{min}}\). Predicate \textit{correct} holds iff \(x \leq c\). The subset of safe states \(S\) is defined by \(S = \{(s, x, y) | 0 < s < \Delta, 0 \leq x \leq c, 0 \leq y \leq s\}\), while the subset of attacked states \(A\) is defined by \(A = \{(s, x, y) | 0 < s < \Delta, c + 1 \leq x \leq S_{\text{min}}, 0 \leq y \leq s\}\). Computation of transition matrix \(P\) is illustrated in Figure 2. In this tree, each edge is labelled by a probability and each leaf represents the state of the cluster. This figure shows all the states that can be reached from state \((s, x, y)\) and the corresponding transition probabilities. Transition probabilities depend on \(i\) the type of event that occurs (join or leave event from the core or the spare set), \(ii\) the type of nodes involved in this operation (honest or malicious), and \(iii\) the ratio of malicious nodes already present in the core set. The probability associated with each one of these states is obtained by summing the products of the probabilities discovered along each path starting from the root to the leaf corresponding to the target state. For instance, the branch on the very right of the tree corresponds to the situation in which cluster \(D_t\) is attacked and one of its malicious core member \(p\) is no more valid. By Property 1, \(p\) leaves \(D_t\), however as \(D_t\) is attacked, the adversary bias the core management procedure by replacing \(p\) with another malicious peer from \(D_t\) spare set. State \((s - 1, x, y - 1)\) is reached.

Modeling and computation of Property 1 is as follows. Let \(d\) be the probability (per unit of time) that the limited lifetime of some peer \(p\) has not expired. Hence \(d\) is homogeneous to a frequency, and \(\Delta t\) \(\Delta t\) represents the probability that the lifetime of a given peer identifier has not expired during \(\Delta t\) units of time. Then the probability that for all the peers belonging to a set of size \(z\) Property 1 holds is equal to \(d^t\).

### 5.4 Initial Distributions

In the experiments conducted for this work, we consider two initial probability distributions. The first one, denoted by \(\alpha^{(1)}\), is such that the cluster starts from a median size and from a failure free state. Namely,

\[
\alpha^{(1)}_{(s, x, y)} = \begin{cases} 
1 & \text{if } (s, x, y) = (\frac{2}{3}, 0, 0) \\
0 & \text{otherwise.}
\end{cases}
\]

The second probability distribution, denoted by \(\alpha^{(2)}\), is such that the cluster starts from a median size and the number of malicious nodes in both the core and spare sets follow a binomial distribution. We have

\[
\alpha^{(2)}_{(s, x, y)} = \binom{S_{\text{min}}}{s} \mu^s (1 - \mu)^{S_{\text{min}} - s} \binom{x}{y} \mu^y (1 - \mu)^{x - y}
\]

for \((\frac{2}{3}, x, y) \in S \cup A\) otherwise.
5.5 Experimental Results

In the remaining of this Section, experiments are conducted with a proportion $\mu = 25\%$ of malicious nodes in the system (cf. Section 2), a core set size $S_{\text{min}} \in \{7, 10\}$, a maximal cluster size $S_{\text{max}} \in \{14, 20\}$, and a number $n$ of clusters in $\{256, \ldots, 4096\}$. Regarding the sojourn time $d$ of nodes, it varies from 0\% (nodes have to leave when they are asked to do so) to 99.9\% (nodes can stay almost infinitely long at the same position in the system, even if they receive a request to leave).

We first show in Figure 3 the distribution of the first instant at which one of the $n$ Markov chains gets absorbed as a function of $m$, for different values of $n$, and starting from different initial states. In Figure 3(a), $S_{\text{min}} = 7$ and $S_{\text{max}} = 14$, which gives a number $N$ of nodes in the system varying in $[1792, 57344]$, while in Figure 3(b), $S_{\text{min}} = 10$ and $S_{\text{max}} = 20$ leading to $N \in [2560, 81920]$. In both Figures 3(a) and 3(b), the distribution of $\Theta_n$ is plotted for the two initial distributions $\alpha^{(1)}$ and $\alpha^{(2)}$ and the sojourn time $d$ of nodes is set to 30\%. As expected, the total number $m$ of join and leave events that need to be triggered before the first cluster splits or merges increases with the size of the system. This is easily explained by the fact that events are uniformly distributed over the clusters and thus the probability for an event to reach a particular cluster decreases as the system size grows. The second observation concerns the very low impact of the initial probability distribution on the distribution of $\Theta_n$. This result, while intuitively surprising, is justified by the fact that $d$ is relatively small ($d = 30\%$) which prevents malicious nodes from staying infinitely long in the same cluster. Consequently, the natural churn of the system overpasses the influence of malicious nodes. Finally, by comparing both Figures 3(a) and 3(b), we observe that a small increase in $\Delta$ (i.e., from $\Delta = 7$ to $\Delta = 10$) significantly augments the first instant at which the first topological change occurs (i.e., a split or merge operation), which is very interesting because it provides a trade-off between the cost implied by the management of core sets and the one implied by topological operations. A deeper investigation into the influence of the initial distribution $\alpha$ on the distribution of $\Theta_n$ is shown in Figures 3(c) and 3(d). The first remark is that for $\alpha = \alpha^{(2)}$, all the curves obtained for the different values of $d$ coincide, meaning that even if malicious nodes may take advantage of longer sojourn times in the same cluster, it takes too much time for them to successfully attack the cluster (due to the design of the robust join and leave operations), and thus they cannot prevent the cluster from splitting and/or merging in response to the natural churn of the system (cf. Section 5.1). Now for $\alpha = \alpha^{(2)}$, the impact of $d$ on the distribution of $\Theta(n)$ is...
significant. Indeed, as clusters start from a failure-prone state, this helps the adversary in reaching more quickly an attacked state, and thus in preventing topological changes from occurring.

Figure 4 depicts the expected proportion of safe clusters after the $m$-th transition for a large number of clusters ($n = 4096$), for a sojourn time $d$ varying from 0% to 99.9%, and for both initial distributions $\alpha$. Let us observe first that when $\alpha = \alpha^{(1)}$ (cf. Figures 4(a) and 4(b)), the expected proportion of safe clusters is very high, even for very large values of $d$. This result combined with the previous observed ones confirms the impact of the join and leave operations on the incapability of the adversary to attack clusters before the first topological change. Indeed, additional graphs (not shown here for space reasons) show that the expected proportion of attacked clusters is very close to 0% which is validated by the relation $E(N_{S_{m}}^{(n)}(m) | \Theta_n > m) = \Pr(\Theta_n > m)$. Now, when clusters start in a failure-prone state, i.e., $\alpha = \alpha^{(2)}$, Figures 4(c) and 4(d) first show that the initial proportion of safe clusters is equal to 75% (which is in accordance with the initial proportion $\mu$ of malicious nodes in the system). Second, if we compare the curves got with $\alpha = \alpha^{(1)}$ and with $\alpha = \alpha^{(2)}$ in presence of a very low induced churn (i.e., $d \geq 90\%$), then we observe that for a given $m$, the expected proportion of safe clusters is larger when the system starts from a failure-prone environment (i.e., $\alpha = \alpha^{(2)}$) than when it starts from a failure-free one. While first intriguing, this result is explained by the fact that the population of malicious nodes is initially larger and as none of the leave events requested for them give rise to the corresponding leave operations, a larger number of events need to be triggered before the first cluster splits or merges. This is a very interesting result as it says that combining robust operations with a very low amount of induced churn is sufficient to keep the clusters in a safe state despite the presence of a large proportion of malicious nodes. Finally, and similarly to Figure 3, increasing $\Delta$ (as shown in Figures 4(b) and 4(d)) augments significantly the time before the first split or merge operation occurs.

6. CONCLUSION

In this paper we have investigated the power of malicious nodes in their capability to subvert large scale dynamic systems. Given an adversarial objective, here maximizing the whole subset of the identifier space malicious nodes succeed in taking control over, we have analytically evaluated the time (in terms of join and leave triggered events) it takes for the adversary to corrupt part of the system prior to the first topological event. Significance of such an analysis was to
determine whether robust operations are capable to prevent pollution from propagating to new clusters, which would ineluctably lead to the progressive subversion of the system. Restricting the analysis to the occurrence of the first topological operation is justified by the fact that topological events occur rarely (i.e., they require a lot of join and leave operations). It is also sufficient to accurately analyze the behavior of the system as by construction of the proposed robust operations, the adversary has no incentive to provoke the triggering of topological operations. Results of the analysis are extremely positive. First they show that by simply increasing the difference between the core and the spare sets, the time that elapses before the first topological operations is significantly augmented. Second they demonstrate that pushing nodes to random positions in the system, even very infrequently, splits adversarial coalitions. Finally, they show that even when the adversary has succeeded in attacking a cluster, the adversary is incentivized to keep their density low. Thus, pollution cannot propagate to safe clusters in their vicinity.

As a future work, we intend to extend the proposed analysis by considering any kind of graphs, imposing solely that the graph formed by correct nodes is connected. One of the issues that we will have to address is the construction of a uniform node sampling algorithm that guarantees that any correct node in the graph has an equal probability to appear in any routing table of any other correct node. The presence of an omniscient adversary makes the issue challenging essentially because such an algorithm cannot rely anymore on the uniform distribution of node identifiers.

7. REFERENCES


