

# Performance Analysis of Large Scale Peer-to-Peer Overlays using Markov Chains

Emmanuelle Anceaume, Bruno Sericola, Romaric Ludinard, Frédéric Tronel

▶ **To cite this version:**

Emmanuelle Anceaume, Bruno Sericola, Romaric Ludinard, Frédéric Tronel. Performance Analysis of Large Scale Peer-to-Peer Overlays using Markov Chains. [Research Report] PI-1963, 2010, pp.18. inria-00546039

**HAL Id: inria-00546039**
**https://inria.hal.science/inria-00546039**

Submitted on 13 Dec 2010

**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.

**Publications Internes de l'IRISA**
**ISSN : 2102-6327**
**PI 1963 – December 2010**

# Performance Analysis of Large Scale Peer-to-Peer Overlays using Markov Chains

Emmanuelle Anceaume* Bruno Sericola** Romaric Ludinard*** Frédéric Tronel****

**Abstract:** In this paper we consider the problem of targeted attacks in large scale peer-to-peer overlays. Targeted attacks aimed at exhausting key resources of targeted hosts to diminish the target node capacity to provide or receive services. To defend the system against such attacks, we rely on clustering and implement induced churn to preserve randomness of nodes identifiers so that adversarial predictions are impossible. We propose robust join, leave, merge and split operations to discourage brute force denial of services and pollution attacks. We show that combining a small amount of randomization in the operations, and adequately tuning the sojourn time of peers in the same region of the overlay allows to decrease the effect of targeted attacks at cluster level, but more importantly to prevent pollution propagation in the whole overlay.

**Key-words:** Clusterized P2P Overlays, Adversary, Churn, Collusion, Markov chains.

---

*Analyse de Performance des réseaux grande échelle à l'aide des Chaînes de Markov*

**Résumé :** *Dans ce rapport nous considérons le problème des attaques ciblées dans les réseaux logiques sous-jacents aux rèseaux grande échelle. La prèsence de comportements malveillants à donner lieu à la construction d'infrastructures logiques robustes à de tels comportements. Cependant, il est communément supposé que la proportion de noeuds malveillants est à tout moment, et n'impote où dans le système est borne et la borne connue. Dans ce rapport nous ne faisons pas de telles hypothèses et nous proposons diverses techniques pour défendre le système contre de telles attaques. Notre construction est validée par analyse.*

**Mots clés :** *Pair-à-pair, adversaire, churn, collusion, Chaînes de Markov*

---

* CNRS UMR 6074 IRISA, emmanuelle.anceaume@irisa.fr, ADEPT INRIA - IRISA
** INRIA Rennes Bretagne-Atlantique, bruno.sericola@inria.fr, DIONYSOS INRIA - IRISA
*** Supélec, romaric.ludinard@supelec.fr, SSIR - Equipe Sécurité des Systèmes d'Information et Réseaux - EA 4039
**** Supélec, frédéric.tronel@rennes.supélec.fr, SSIR - Equipe Sécurité des Systèmes d'Information et Réseaux - EA 4039

©IRISA – Campus de Beaulieu – 35042 Rennes Cedex – France – +33 2 99 84 71 00 – www.irisa.fr

# 1 Introduction

In this paper we consider the problem of targeted attacks in large scale peer-to-peer systems. Targeted attacks aimed at exhausting key resources of targeted hosts (e.g., bandwidth, CPU processing, TCP connection resources) to diminish the target node capacity to provide or receive services [1], at preventing data indexed at targeted nodes from being discovered and retrieved by poisoning their routing tables, or simply at rerouting or dropping messages addressed to targeted nodes. Such malicious behaviors have led to the proposition of malicious-resilient overlay systems (e.g., [2, 3, 4]). In all these systems, it is assumed that at any time, and anywhere in the overlay, the proportion of compromised peers is bounded and known. Unfortunately, targeted attacks violate such an assumption, and thus additional mechanisms are needed. It has been shown that peer-to-peer overlays can survive these attacks only if malicious nodes are not able to isolate honest nodes within the system. This is achieved by *i)* preserving randomness of nodes identifiers, so that malicious nodes are not able to predict what will be the topology of the network after a given sequence of operations [5], and *ii)* limiting the period of time where nodes can stay at the same position in the overlay. Induced churn has been shown to be a fundamental ingredient to preserve randomness. Churn is classically defined as the rate of turnover of peers in the system [6], and thus induced churn refers to the general idea of forcing peers to move within the system. Several strategies based on this principle have been proposed. Most of them are based on locally induced churn. However either they have been proven incorrect or they involve a level of complexity that is too high to be practically acceptable [5]. Some other strategies, based on globally induced churn, force each node to periodically leave and re-join the system. This may be enforced by imposing limited lifetime on each node in the system. If not properly handled, these solutions keep the system in an unnecessary hyper-activity, which increases accordingly the impact of churn.

In the present work, we investigate adversarial strategies that aim at isolating honest nodes in cluster based overlays, and we present an analytical study of the long term behavior of the system. Our analysis shows that *i)* by limiting the sojourn time of nodes at the same position in the overlay and *ii)* by introducing randomness in the operations of the overlay, pollution attacks are severely reduced at cluster level and do not propagate to the whole overlay. A preliminary work [7] investigates adversarial strategies in the specific context where the sequence of join and leave events is well interleaved. In this paper we extend this preliminary work to a general context in which clusters size varies with the churn and thus can undergo merge and split operations. To the best of our knowledge, this is the first work that has conducted such a study.

The remainder of this paper is as follows: In Section 2, we present existing works that focus on making structured-based overlays robust against attacks. In Section 3 we briefly describe the main features of cluster-based overlays, and present the assumptions made in this work. Section 4 describes the robust operations implemented in the overlay. Then, in Section 5, we specify the strategy the adversary adopts to perform targeted attacks in the system. The adversarial behavior is modeled and its impact at both cluster level and at the overlay level are respectively studied in Sections 6 and 7.

# 2 Related Work

Different approaches have been proposed to face malicious behavior, 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 peers 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 [8]. Such an approach has been implemented into several overlays (e.g., [9, 10, 11]). Another defense against those attacks comes from the observation that during eclipse attacks, the degree of attackers is much higher than the average degree of peers in the overlay. Addressing such attacks consists in bounding peers degrees. Singh et al. [12] propose to anonymously audit peers to continuously check the bounded degree of peers. Results of their experimentation show that audit efficiency depends on both the audit rate and the stability of the overlay. This makes their solution is effective in an overlay with low to moderately high churn. More generally, to prevent messages from being misrouted or dropped, the seminal works on DHT routing security by Castro et al. [8] and Sit and Morris [13] 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., [2, 4, 14]). In all these above cited works, it is assumed that at any time, and anywhere in the overlay, the proportion of compromised peers is bounded and *a priori* known. It allows powerful building blocks such as Byzantine tolerant agreement protocols to be used among peers subsets [2, 4]. When such an assumption does not hold, additional mechanisms are needed. Awerbuch et al [5] propose the *Cuckoo&flip* strategy. This strategy consists in introducing local induced churn (i.e., forcing a subset of peers to leave the overlay) upon each join and leave operation. This strategy prevents malicious peers from predicting what is going to be the state of the overlay after a given sequence of join and leave operations. Subsequently to this work, experiments have been conducted to check the feasibility of global induced churn. These experiments assume that the overlay is populated by no more than $\mu = 25\%$ of compromised peers [15], or that the topology of the overlay is static [7].

# 3 Overlays Networks

An overlay network is a virtual network built on top of a physical network. Nodes of the overlay, usually called *peers*, communicate among each other along the edges of the overlay by using the communication primitives provided by the underlying network (e.g., IP network service). The algorithms that peers use to choose their neighbors and to route messages define the overlay topology. The topology of unstructured overlays conforms with random graphs, i.e., relationships among peers are mostly set according to a random process which makes joining and leaving operations constraint free. Data can be located in any peer thereby imposing flooding or random walk techniques to retrieve them. Randomly placing data in the network guarantees that attacks against data are difficult to mount. On the other hand, it requires a linear number of queries to be retrieved which is definitively not scalable [16]. This scalability issue can be circumvented at the price of strong restrictions on churn [17]. Structured overlays (also called Distributed Hash Tables (DHTs)) build their topology according to structured graphs (e.g., hypercube, torus). For most of them, the following principles hold: each peer is assigned a unique random identifier from an $m$-bit identifiers space. Identifiers (denoted IDs in the following) are derived by applying some standard cryptographic one-way hash function on peers intrinsic characteristics (e.g., IP address). The value of $m$ (128 for the standard MD5 hash function) is large enough to make the probability of identifiers collision negligible. The identifier space is partitioned among all the peers of the overlay. Peers self-organize within the structured graph according to a distance function $D$ based on peers IDs (e.g., two peers are neighbors if their IDs share some common prefix), plus possibly other criteria such as geographical distance. Each data is assigned a unique identifier, called *key*, selected from the same $m$-bit identifiers space. Each peer $p$ owns a fraction of all the data items of the overlay. The mapping derives from the distance function $D$. In the following, we will use the term peer (or key) to refer to both the peer (or key) and its $m$-bit representation.

Following the seminal work of Plaxton et al [18], diverse DHTs have been proposed (e.g., [9, 10, 11, 19]). All these DHTs have proven to be highly satisfactory in terms of efficiency and scalability (i.e., their key-based routing interface guarantees operations whose complexity in messages and latency usually scale logarithmically with system size). However, in presence of adversarial behavior, relying on single peers to ensure the system connectivity and the correct retrieval of data is clearly not sufficient, as by holding a logarithmic number of peer IDs the adversary can in a linear number of trials disconnect some target from the overlay. On the other hand, by having peers gathered into quorums or clusters, one can introduce the unpredictability required to deal with Byzantine attacks through randomized algorithms. This has led to cluster-based structured overlay networks.

## 3.1 Cluster-based Structured Overlays Networks

Cluster-based structured overlay networks are such that clusters of peers substitute for peers at the vertices of the structured graph. Each vertex of the structured graph is composed of a set or *cluster* of peers. Peers join the clusters according to a given distance metric $D$. For instance in PeerCube [4], peer $p$ joins the (unique) cluster whose label is a prefix of $p$'s identifier, while in eQuus [20], $p$ joins the (unique) cluster whose members are geographically the closest to $p$. Clusters in the overlay are uniquely labelled. Size of each cluster is both lower and upper bounded. The lower bound, named $C$ in the following, usually satisfies some constraint based on the

assumed failure model. For instance $C \geq 4$ allows Byzantine tolerant agreement protocols to be run among these $C$ peers despite the presence of one Byzantine peer [21]. The upper bound, that we will call $S_{max}$, is typically in $\mathcal{O}(\log U)$ where $U$ is the current number of peers in the overlay, to meet scalability requirements. Once a cluster size exceeds $S_{max}$, this cluster `splits` into two smaller clusters, each one populated with the peers that are closer to each other according to distance $D$. Once a cluster undershoots its minimal size $C$, this cluster `merges` with the closest cluster in its neighborhood.

In the present work we assume that at cluster level, peers are organized as core and spare members. 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 $C$. Spare members are the complement number of peers in the cluster. Size $s$ of the spare set is such that $s \leq \Delta$ where $\Delta = S_{max} - C$. In contrast to core members, spare members are not involved in any of the overlay operations. Rationale of this classification is two-fold: first it limits the management overhead caused by the natural churn present in typical overlay networks through the spare set management. Second it allows to introduce the unpredictability required to deal with Byzantine attacks through a randomized core set generation algorithm as shown in the sequel. In the following we assume that join and leave events have an equal chance to occur in any cluster.

### 3.2 Model of the Adversary

A fundamental issue faced by any practical open system is the inevitable presence of peers that try to manipulate the system by exhibiting undesirable behaviors [13]. Such peers are classically called malicious or Byzantine. Malicious peers can devise complex strategies to prevent peers from discovering the correct mapping between peers and data keys. They can mount *Sybil attacks* [22] (i.e., an attacker generates numerous fake peers to pollute the system), they can do *routing-table poisoning* (also called *eclipse attacks* [8, 13]) by having honest peers redirecting outgoing links towards malicious ones, or they can simply drop or re-route messages towards other malicious peers. They can magnify their impact by colluding and coordinating their behavior. We model these strategies through a strong adversary that controls these malicious peers. 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 peers in the whole network. Note that in the following we make a difference between the whole network and the overlay. The network encompasses all the peers that at some point may participate to the overlay (i.e. $2^m$ peers), while the overlay contains at any time the subset $U$ of participating peers. Thus, while $\mu$ represents the assumed fraction of malicious peers in the network, the goal of the adversary is to populate some parts of the overlay with a larger fraction of malicious peers in order to subvert the correct functioning of the overlay. Finally, a peer which always follows the prescribed protocols is said *honest*. Note that honest peers cannot *a priori* distinguish honest peers from malicious ones.

### 3.3 Security Schemes

We assume the existence of a public key cryptography scheme that allows each peer to verify the signature of each other peer. We also assume that correct peers never reveal their private keys. Peers IDs and keys (private and public) are acquired via a registration authority. When describing the protocols, we ignore the fact that messages are signed, and recipients of a message ignore any message that is not signed properly. We also use cryptographic techniques to prevent a malicious peer from observing or unnoticeably modifying a message sent by a correct peer. However a malicious peer has complete control over the messages it sends and receives.

### 3.4 Limited Sojourn Time and Random Distribution of IDs

As said in the Introduction, it has been shown by Awerbuch and Scheideler [16] 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 firstly,

the guarantee that the distribution of peers identifiers is random, and secondly that peers cannot stay forever at the same position in the system [5]. Both properties have been analytically proven assuming that the region size is kept constant [7].

To implement both limited sojourn time of the nodes at the same place in the overlay and unpredictable identifier assignment in a cluster-based overlay, we propose to proceed as follows: Peers identifiers are initially generated based on certificates acquired at trustworthy Certification Authorities (CAs). Initial identifiers (denoted $ID^0$) are generated as the result of applying a hash function $\mathcal{H}$ to some of the fields of a X.509 [23] certificate. To enforce all peers, including malicious ones, to leave and rejoin the system from time to time, we add the creation date $t_0$ to the fields that appear in the peer certificate that will be hashed to generate the peer identifier (note that by the properties of hash functions, this guarantees that peers identifiers are unpredictable). The incarnation number limits the lifetime of identifiers. The current incarnation $k$ of any peer is given by the following expression $k = \lceil (t - t_0)/L \rceil$, where $t_0$ is the initial validity time of the peer's certificate, $t$ is the current time, and $L$ is the length of the lifetime of each peer's incarnation. Thus, the $k^{th}$ incarnation of a peer $p$ expires when $p$ local clock reads $t_0 + kL$. At this time $p$ must rejoin the system using its $(k+1)^{th}$ incarnation. The $t_0$ parameter is one of the fields in the peer's certificate and since certificates are signed by the CA, it cannot be unnoticeably modified by a malicious peer. Moreover, a certificate commonly contains the public key of the certified entity. This way, messages exchanged by the peers can be signed using the sender private key, preventing malicious peers from unnoticeably altering messages originated from other peers in the system. Messages must contain the certificate of their issuer, so as to allow recipients to validate them [1].

Therefore, at any time, any peer can check the validity of the identifier of any other peer $q$ in the system, by simply calculating the current incarnation $k$ of the other peer $q$ and generating the corresponding identifier. Specifically, the current identifier of peer $q$, denoted in the following as $ID_q$, is calculated by hashing $q$ initial identifier ($ID_q^0$) with the current incarnation $k$ of $q$, i.e., $ID_q = \mathcal{H}(ID_q^0 \times k)$.

**Property 1 (Limited Sojourn Time)** *Let $\mathcal{D}$ be some cluster of the system and $p$ some peer in the overlay. Then $q$ belongs to $\mathcal{D}$ at time $t$ if only if $ID(q)$ matches the label of $\mathcal{D}$ according to distance $D$ (we say that $q$ is valid for $\mathcal{D}$).* ■

If some peer detects that the ID of one of its neighbors $q$ is not valid then it cuts its connection with $q$. Note that node $q$ may re-join the overlay by triggering a `join` operation at cluster $\mathcal{D}'$ such that $ID(q)$ is valid for $\mathcal{D}'$. Note that because clocks are loosely synchronized, it is possible that a correct peer is still using its ID for incarnation $k$ when other correct peers would expect it to be in incarnation $k+1$. To mitigate this problem, we assume that any correct peer may have two subsequent valid incarnation numbers, for a fixed grace window $W$ of time that encompasses the expiration time of an incarnation number ($W$ is the maximum deviation of the clocks of any two correct peers). More precisely, at any time $t$, both incarnation $k$ and $k'$ are valid, where $k = \lceil (t - W/2 - t_0)/L \rceil$, and $k' = \lceil (t + W/2 - t_0)/L \rceil$. Note that this means that although at any time $t$ each peer $p$ has a single incarnation number that it uses to define its current ID, other peers calculate two possible incarnation numbers for $p$. These are frequently equal, but may differ when $p$'s local time is close to the expiration time of its current/last incarnation.

# 4 Operations of the Overlay

To protect the system against the presence of malicious peers in the overlay, we propose to take advantage of peers role separation at cluster level to design robust operations. Briefly, to discourage brute force denial of service attacks, `joining` peers are inserted in their cluster as spare members and not as core members. Beyond keeping churn management at cluster level, this impedes the adversary from designing deterministic strategies to join the core set in order to manipulate the overlay operations. To impede 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

[1] Note that there are means to optimize this procedure as for example by exchanging certificates during an initialization phase. However this is out of the scope of our paper

peers, `leave` operations for peers in the core set give rise to the complete or partial renewal of this set through a randomization maintenance process. As both `merge` and `split` operations induce topological changes (i.e. such operations involve routing table updates in the neighborhood of the cluster that triggers these operations), and more importantly may have an influence on the subset of the identifier space the adversary may gain control over (a merge operation doubles the subset of the identifier space a cluster is responsible for, while a split operation divides it per two), these operations have been designed so that the adversary has, in expectation, no interest to trigger them. Briefly, the outcome of a `split` operation is the partial randomization of the two created core sets, while the `merge` operation gives rise to a change in the leadership of the new created cluster. Specifically,

- `join`(p): when a peer $p$ `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 peer $p$ `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 $k-1$ randomly chosen members of the core set with $k$ peers randomly chosen from the whole cluster, where $1 \leq k \leq C$ (recall that $C$ is the size of the core set, cf. Section 3.1).
- `split`($\mathcal{D}$): when a cluster $\mathcal{D}$ has reached the conditions to `split` into two smaller clusters $\mathcal{D}'$ and $\mathcal{D}''$, core sets of both $\mathcal{D}'$ and $\mathcal{D}''$ are built. Priority is given to core members of $\mathcal{D}$ and completion is done with randomly chosen spares in $\mathcal{D}$. This random choice is handled through a Byzantine-tolerant consensus run among core members of $\mathcal{D}$. Spares members of $\mathcal{D}'$ (resp. $\mathcal{D}''$) are populated with the remaining spares members of $\mathcal{D}$ that are closer to $\mathcal{D}'$ than to $\mathcal{D}''$ (resp. closer to $\mathcal{D}''$ than to $\mathcal{D}'$).
- `merge`($\mathcal{D}$): when some cluster $\mathcal{D}'$ has reached the conditions to `merge` (i.e., its spare set is empty), it merges with the closest cluster $\mathcal{D}''$ to $\mathcal{D}'$. The created cluster $\mathcal{D}$ is composed by a core set whose members are the core set members of $\mathcal{D}''$ and by a spare set whose members are the union of the spare members of $\mathcal{D}''$ and the core set members of $\mathcal{D}'$.

These four operations make up the overlay protocol. In the following $protocol_k$ will refer to as these four operations with $1 \leq k \leq C$ the amount of randomization of the core view maintenance procedure of the `leave` operation.

# 5 Specification of the Adversarial Behavior

Based on the operations described in the previous section, we investigate how malicious peers could proceed to compromise correctness of a targeted cluster. Clusters correctness is jeopardized as soon as a quorum of its core members are malicious. It is well known that 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 [21]. In our context, cluster $\mathcal{D}$ is said to be *polluted* if its core set is populated by more than a quorum $c$ of malicious peers where $c = \lfloor (C-1)/3 \rfloor$ malicious peers (recall that $C$ is the size of the core set of any cluster $\mathcal{D}$). Otherwise, this cluster $\mathcal{D}$ is said *safe*.

## 5.1 Increasing Global Representation of Malicious Identifiers

As a consequence of assigning an initial unique random ID from an $m$-bit identifier space to peers and of periodically pushing peers to random regions in the overlay, a primary strategy for the adversary to gain the core quorum of targeted clusters is to maximize the number of malicious peers that sit in the whole overlay. By doing this, the adversary augments the likelihood for its malicious peers to join targeted clusters. For those peers which cannot immediately enter targeted clusters (because their current IDs do not match the targeted clusters label), but rather join different clusters, the goal of the adversary is to pollute these clusters as well. This augments the

subset of identifiers space the adversary has gained control over, and thus empowers it to progressively surround the targeted clusters.

From an operational point of view, peers cannot `join` a cluster as core members (cf Section 4). For peer $p$ to be inserted into the core set of $\mathcal{D}$, $p$ must first `join` the spare set of $\mathcal{D}$, and then must be chosen as a core member by the core set maintenance process run subsequent to the departure of one of its core member. Of course, $p$ identifier must also be valid (in the sense of Property 1).

Note that the adversary may also trigger `leave` operations for malicious peers in the core so as to increase the global representation of malicious identifiers in core sets. Specifically if, by having a malicious peer `leaving` the core set of a cluster, the likelihood that the outcome of the core set maintenance process strictly increases the number of malicious peers in the renewed core set, then the adversary triggers a `leave` operation. Formally,

**Rule 1 (Adversarial `Leave` Strategy)** *Let $\mathcal{D}$ be a cluster such that at time $t$ its core set $\mathcal{C}$ contains $0 < x \leq c$ valid malicious peers and its spare set contains $y > 1$ valid malicious peers. At time $t$ the adversary triggers a `leave`($p$) operation for malicious peer $p$ in $\mathcal{C}$ if, for a given small positive threshold $\nu$*

$$\sum_{j=x+1}^{x-1+\min(k,y)} \mathbb{P}\left\{ \begin{array}{c} \textit{exactly } j \textit{ malicious peers} \in \mathcal{C} \\ \textit{after the } \texttt{leave} \textit{ operation} \end{array} \right\} > 1 - \nu. \tag{1}$$

*Recall that $k$ is the amount of randomization of the `leave` operation. Note that for $k = 1$, Relation (1) is never satisfied. Thus in this specific case, there is no incentive for malicious peers to trigger voluntary `leaves`. For $k > 1$, malicious peers collude together to force the one among them (whose ID will expire the soonest) to `leave` the core. As will show the experiments, decreasing the amount of randomization of the `leave` operation provides the best strategy against targeted attacks.* ■

Once the adversary has succeeded in polluting a cluster $\mathcal{D}$, his goal is to minimize the likelihood that $\mathcal{D}$ switches back to a safe state. Switching to a safe state may occur subsequent to either the core maintenance procedure, the `merge`, or the `split` operations. The two latter cases are detailed in the following section. Regarding the former case, the adversary can bias the core set maintenance procedure by replacing the left peer with a (valid) malicious peer from the spare set if any. On the other hand, if the spare set does not contain any malicious peers then the adversary has no choice other than choosing a honest peer.

### 5.2 Decreasing the Occurrence of Topological Operations

So far we have seen that the adversary may trigger `leave` operations for its malicious peers if this increases with high probability the population of malicious peers in the core set. However, the adversary will trigger such departures only if this does not lead the cluster to `merge` with another cluster. Indeed, by construction of the `merge` operation (cf. Section 4), when $\mathcal{D}$ core members trigger a `merge` with their closest neighbor $\mathcal{D}'$ then all $\mathcal{D}$ members are pushed to the spare set of the new created cluster $\mathcal{D}''$ while core members of $\mathcal{D}'$ keep their status of core members in $\mathcal{D}''$. This clearly deters the adversary from triggering `merge` operations.

We have also seen that to gain the control of clusters, the strategy of the adversary is to maximize the number of malicious peers in both the core and the spare sets of any cluster. However once a cluster is polluted, the adversary has no interest to let this cluster grow in such a way that this cluster will undergo a `split` operation. Indeed, the outcome of a `split` operation cannot increase the subset of identifiers space the adversary has gained control over—at best, it keeps it the same. Thus when a polluted cluster is close to `split`, the adversary acts so that no `join` operations are triggered. Note that the adversary can prevent honest peers from joining $\mathcal{D}$ whenever $s > 1$. This guarantees that $\mathcal{D}$ will not grow because of honest peers, while ensuring that $\mathcal{D}$ will not undergo a `merge` operation as much as possible. Specifically,

**Rule 2 (Adversarial `Join` Strategy)** *Let $\mathcal{D}$ be a cluster such that at time $t$ its core set contains $\ell > c$ valid malicious peers. Any join event issued by peer $q$ and received at $\mathcal{D}$ at time $t$ is discarded (i.e., the associated `join` operation is not triggered) if ($q$ is honest and $s > 1$) or ($s = \Delta - 1$). Recall that $\Delta$ represents the maximal size of the spare set (c.f. Section 3.1).* ■

A possible implementation of Rule 2 is as follows: upon receipt of a join event from an honest peer $q$, the adversary asks his malicious core members to positively acknowledge $q$ so that $q$ does not detect that $\mathcal{D}$ is polluted, however the associated `join` operation is not triggered.

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 either Property 1 does not hold or Rule 1 holds, and second by having the maximal number of malicious peers join the system except if Rule 2 holds.

# 6 Modeling the Adversarial Strategy

The evolution of any given cluster $\mathcal{D}$ follows the overlay protocol $protocol_k$ (cf. Section 4) and the strategy of the adversary. To take into account this strategy, we make a difference between join (resp. leave) events and `join` (resp. `leave`) operations. Join and leave events targets are distributed uniformly at all the peers of the overlay, and thus in particular, at malicious peers. On the other hand, upon receipt of such an event (in particular a leave event), a malicious peer may ignore it by simply not triggering the corresponding `leave` operation (cf. Rule 2).

We model the effect of join and leave events 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 malicious peers in both the core set and the spare set of cluster $\mathcal{D}$. The state space $\Omega$ of $X$ is defined by $\Omega = \{(s, x, y) \mid 0 < s < \Delta,\ 0 \leq x \leq C,\ 0 \leq y \leq s\}$. For $n \geq 1$, the event $X_n = (s, x, y)$ means that, after the $n$-th transition (i.e., the $n$-th join or leave event), 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 $y$. In the remaining of the paper, the initial probability distribution of $X$ is denoted by $\alpha$. The transition probability matrix $M$ of $X$ depends on both $protocol_k$ and on the adversarial behavior. This matrix is detailed below.

We define a state as *polluted* if in this state the core set contains more than $c = \lfloor (C-1)/3 \rfloor$ malicious peers. Conversely, a state that is not polluted is said to be *safe*.

## 6.1 Modeling Protocol$_k$

The subset of safe states, denoted by $S$, is defined by $S = \{(s, x, y) \mid 0 < s < \Delta,\ 0 \leq x \leq c,\ 0 \leq y \leq s\}$, while the subset of polluted states, denoted by $P$, is defined by $P = \{(s, x, y) \mid 0 < s < \Delta,\ c+1 \leq x \leq C,\ 0 \leq y \leq s\}$. The system alternates between safe and polluted states until entering closed states. Closed states represent the logical disappearance of cluster $\mathcal{D}$ from the graph. This occurs when either $\mathcal{D}$ splits into two smaller clusters (i.e., its spare set has reached its maximal size $\Delta$) or $\mathcal{D}$ merges with its closest neighbor (i.e., the size of its spare set has shrunk to 0). Three sets of closed states exist. The set of *safe merge* closed states $A_S^m$ defined as $A_S^m = \{(s, x, y) \mid (s = 0) \wedge (0 \leq x \leq c)\}$, the set of *safe split* closed states $A_S^\ell$ defined as $A_S^\ell = \{(s, x, y) \mid (s = \Delta) \wedge (0 \leq x \leq c)\}$, and the set of *polluted merge* closed states $A_P^m$ defined as $A_P^m = \{(s, x, y) \mid (s = 0) \wedge (c+1 \leq x \leq C)\}$. Note that by Rule 2 the states such that $s = \Delta$ and $c+1 \leq x \leq C$ are not reachable because the adversary strategizes to prevent a polluted cluster from triggering a `split` operation (cf. Section 5.2). As a consequence the set of *polluted split* closed states do not exist. Matrix $P$ is partitioned in a manner conformant to the decomposition of $\Omega = S \cup P \cup A_S^m \cup A_S^\ell \cup A_P^m$

$$M = \begin{pmatrix} M_S & M_{SP} & M_{SA_S^m} & M_{SA_S^\ell} & M_{SA_P^m} \\ M_{PS} & M_P & M_{PA_S^m} & M_{PA_S^\ell} & M_{PA_P^m} \\ 0 & 0 & M_{A_S^m} & 0 & 0 \\ 0 & 0 & 0 & M_{A_S^\ell} & 0 \\ 0 & 0 & 0 & 0 & M_{A_P^m} \end{pmatrix}$$

where $M_{UV}$ is the sub-matrix of dimension $|U| \times |V|$ containing the transitions from states of $U$ to states of $V$, with $U, V \in \{S, P, A_S^m, A_S^\ell, A_P^m\}$. We simply write $M_U$ for $M_{UU}$. Note that $A_S^m$, $A_S^\ell$, and $A_P^m$ are absorbing classes. In the same way, the initial probability distribution $\alpha$ is partitioned by writing $\alpha = (\alpha_S \ \ \alpha_P \ \ \alpha_{A_S^m} \ \ \alpha_{A_S^\ell} \ \ \alpha_{A_P^m})$, where sub-vector $\alpha_U$ contains the initial probabilities of states $U \in \{S, P, A_S^m, A_S^\ell, A_P^m\}$. Figure 1 shows the states partition of process $X$.

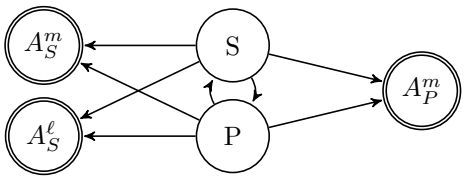



Figure 1: Aggregated view of Markov chain $X$. Transient safe and polluted states are respectively represented by $S$ and $P$. States $A_S^m$, $A_S^\ell$ and $A_P^m$ represent the closed safe and polluted states.

Computation of transition matrix $M$ 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 reached from state $(s, x, y)$ and the corresponding transition probabilities. Transition probabilities depend on *i)* the type of operation that occurred (join or leave operation from the core or the spare set), *ii)* the type of peers involved in this operation (honest or malicious), and *iii)* the ratio of malicious peers 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 leftmost branch of the tree corresponds to the scenario in which some malicious peer wishes to join the polluted cluster $\mathcal{D}$. By Rule 2, this peer successfully joins the cluster, leading to state $(s+1, x, y+1)$. Now if we consider the rightmost branch in the tree, this represents the situation in which cluster $\mathcal{D}$ is polluted and one of its malicious core member $p$ is no more valid. By Property 1, $p$ leaves $\mathcal{D}$, however as $\mathcal{D}$ is polluted, the adversary bias the core management procedure by replacing $p$ with another malicious peer from $\mathcal{D}$ spare set. State $(s-1, x, y-1)$ is reached. Derivation of the transition probability matrix $M$ is presented in Figure 3.

Modeling and computation of Property 1 and Rule 1 are achieved as follows. Regarding Property 1, let $d$ be the probability that at time $n \geq 1$ the sojourn time of some peer $p$ has not expired. Then the probability that for all the peers belonging to a set of size $z$ Property 1 holds is equal to $d^z$. Regarding Rule 1, let $q(k, \ell, u, v)$ be the probability of getting $u$ red balls when $k$ balls are drawn without replacement from an urn containing $v$ red balls and $\ell - v$ white balls. We have $q(k, \ell, u, v) = \binom{v}{u}\binom{\ell-v}{k-u} / \binom{\ell}{k}$. $q(.)$ is referred to as the hypergeometric distribution. Let $(s, x, y)$ be the current state of the Markov chain associated to cluster $\mathcal{D}$. Then Rule 1 holds if the chain enters one of the following states $(s-1, x+1, y-2), \ldots, (s-1, x', y')$, with $x' = x + y - 1$ and $y' = 0$ if $k \geq x + y - 1$ otherwise $x' = k$ and $y' = x + y - 1 - k$, right after the voluntary departure of one malicious valid core member with probability $1 - \nu$. From the `leave` operation Relation (1) writes as

$$\sum_{i=i_0}^{i_{\max}} \sum_{j=i+2}^{j_{\max}} q(k-1, C-1, i, x-1) q(k, s+k-1, j, y+i) > 1 - \nu. \tag{2}$$

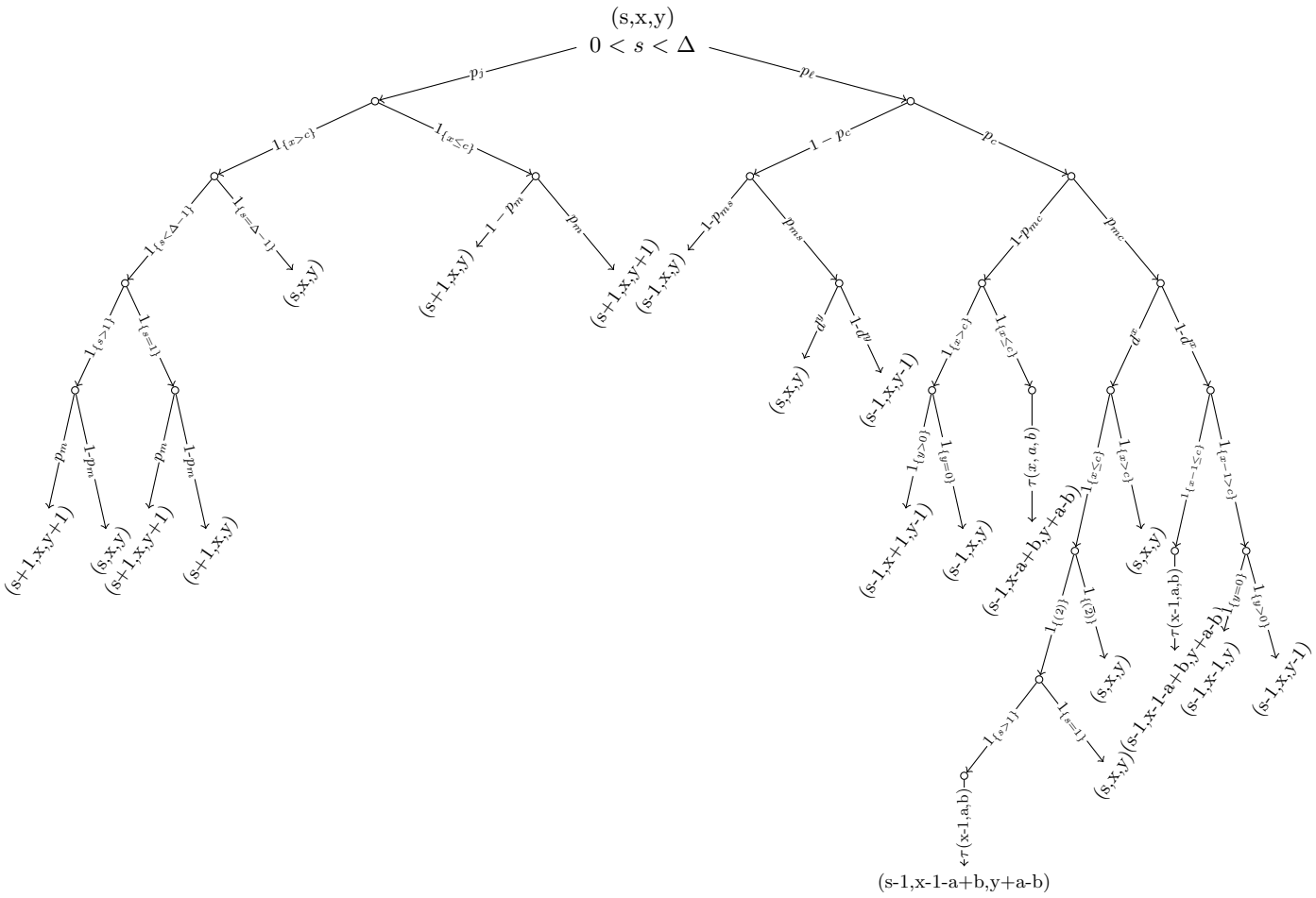
where $i_0 = k - 1 - \min(k-1, C-x)$, $i_{\max} = \min(k-1, x-1)$ and $j_{\max} = \min(k, y+i)$. In Figure 2, notation "$1_{\{(2)\}}$" means that Relation (2) holds.

## 6.2 Study of a Cluster Behavior

In this section, we study the behavior of a cluster according to the power of the adversary in terms of resources (i.e., $\mu$ value), the frequency of the induced churn, and the amount of randomization $k$ introduced in the `leave` operation.

### 6.2.1 Initial Distributions

In the experiments conducted for this work, we consider two initial distributions. The first one, which we denote by $\beta$, consists in assuming that the initial size of the spare set (denoted as $s_0$) is uniformly distributed on $\{1, \ldots, \Delta - 1\}$. The initial number $C_0$ of malicious peers in the core set and the one $S_0$ in the spare set both follow a binomial distribution. The initial state $X_0$ is thus defined by $X_0 = (s_0, C_0, S_0)$. Assuming that $C_0$ and $S_0$ are independent, we get for $x = 0, \ldots, C$ and $y = 0, \ldots, s_0$




| Probabilities | Value | Significance of the probability |
|---|---|---|
| $p_j$ (resp. $p_\ell$) | $1/2$ | join (resp. leave) event probability |
| $p_c$ | $C/(C+s)$ | probability for a peer to belong to the cluster core set |
| $p_m$ | $\mu$ | probability that the joined peer is malicious |
| $p_{mc}$ | $x/C$ | probability for a core member to be malicious |
| $p_{ms}$ | $y/s$ | probability for a spare member to be malicious |
| | $1-d^x$ (resp. $1-d^y$) | probability that Property 1 is satisfied in the core (resp. spare) set |
| $1_{\{A\}}$ | 1 if condition $A$ is true, 0 otherwise | represents the indicator function |
| $\tau(x,a,b)$ | $q(k-1,C-1,a,x)q(k,s+k-1,b,y+a)$ with $q(k,\ell,u,v) = \binom{v}{u}\binom{\ell-v}{k-u} \,/\, \binom{\ell}{k}$ | probability of building the core (resp. spare) set with $x-a+b$ (resp. $y-b+a$) malicious peers where $\max(0,k-1-(C-1-x)) \leq a \leq \min(x,k-1)$, and $\max(0,k-(s+k-1-(y+a)) \leq b \leq \min(y+a,k)$ |

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

For all $s \in \{1, \ldots, \Delta - 1\}$, for all $x \in \{0, \ldots, C\}$ and for all $y \in \{0, \ldots, s\}$, we have:

$$
\begin{aligned}
p_{(s,x,y),(s,x,y)} &= p_j 1_{\{x>c\}} \left(1_{\{s=\Delta-1\}} + 1_{\{1<s<\Delta-1\}}(1-p_m)\right) \\
&+ p_\ell \left( d^y \frac{y}{C+s} + d^x \frac{x}{C+s} \left(1_{\{x>c\}} + 1_{\{x\leq c\}} \left(1_{\{(\overline{2})\}} + 1_{\{s=1\}} 1_{\{(2)\}}\right)\right)\right) \\
p_{(s,x,y),(s+1,x,y)} &= p_j \left(1_{\{x>c\}} 1_{\{s=1\}} + 1_{\{x\leq c\}}\right)(1-p_m) \\
p_{(s,x,y),(s+1,x,y+1)} &= p_j \left(1_{\{x>c\}} 1_{\{s<\Delta-1\}} \left(1_{\{s>1\}} + 1_{\{s=1\}}\right) + 1_{\{x\leq c\}}\right) p_m \\
p_{(s,x,y),(s-1,x,y-1)} &= p_\ell \left( (1-d^y) \frac{y}{C+s} + (1-d^x) \frac{x}{c+s} \left(1_{\{x-1\leq c\}} \tau(x-1,a,b) + 1_{\{x-1>c\}} 1_{\{y>0\}}\right)\right) \text{ for } b-a=1 \\
p_{(s,x,y),(s-1,x-1,y)} &= p_\ell (1-d^x) \frac{x}{C+s} \left(1_{\{x-1\leq c\}} \tau(x-1,a,b) + 1_{\{x-1>c\}} 1_{\{y=0\}}\right) \text{ for } b-a=0 \\
p_{(s,x,y),(s-1,x,y)} &= p_\ell \left( \frac{s-y}{C+s} + \frac{C-x}{C+s} \left(1_{\{x>c\}} 1_{\{y=0\}} + 1_{\{x\leq c\}} \tau(x,a,b)\right)\right) \text{ for } b-a=0 \\
p_{(s,x,y),(s-1,x+1,y-1)} &= p_\ell \frac{C-x}{C+s} \left(1_{\{x>c\}} 1_{\{y>0\}} + 1_{\{x\leq c\}} \tau(x,a,b)\right) \text{ for } b-a=1 \\
p_{(s,x,y),(s-1,x-a+b,y+a-b)} &= p_\ell \frac{C-x}{C+s} 1_{\{x\leq c\}} \tau(x,a,b) \\
\text{for } b-a &\notin \{0,1\} \\
\text{and } x-a+b &\geq x - \min(x,k-1) + k - \min(s-y+\min(C-1-\min(x,k-1), k-1-\min(x,k-1)), k) \\
\text{and } x-a+b &\leq \min(x, C-k) + \min(k, x+y-\min(x, C-k)) \\
p_{(s,x,y),(s-1,x-1-a+b,y+a-b)} &= p_\ell \frac{x}{C+s} \left(d^x 1_{\{x\leq c\}} 1_{\{(2)\}} 1_{\{s>1\}} + (1-d^x) 1_{\{x-1\leq c\}}\right) \tau(x-1,a,b) \\
\text{for } b-a &\notin \{0,1\} \\
\text{and } x-a+b &\geq x - 1 - \min(x-1,k-1) + k - \min(s-y+\min(C-\min(x-1,k-1), k-1-\min(x-1,k-1)), k) \\
\text{and } x-a+b &\leq \min(x-1, C-k) + \min(k, x-1+y-\min(x-1, C-k)).
\end{aligned}
$$

In all other cases, transition probabilities are null.

Figure 3: Transition probabilities of the Markov chain of process $X$ starting from state $(s,x,y)$ for all $x \in \{0,\ldots,C\}$ and for all $y \in \{0,\ldots,s\}$ with $0 < s < \Delta$.

| | $\mu = 0\%$ | | | $\mu = 10\%$ | | | $\mu = 20\%$ | | | $\mu = 30\%$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $d$ | 0.95 | 0.99 | 0.999 | 0.95 | 0.99 | 0.999 | 0.95 | 0.99 | 0.999 | 0.95 | 0.99 | 0.999 |
| $E(T_S^{(1)})$ | 12.0 | 12.0 | 12.0 | 12.09 | 12.08 | 12.08 | 11.88 | 11.84 | 11.83 | 11.54 | 11.48 | 11.47 |
| $E(T_P^{(1)})$ | 0.0 | 0.0 | 0.0 | 0.15 | 2.6 | 1518 | 1.14 | 699.7 | 511810822. | 5.96 | 12597. | 9299884149 |

Table 1: $E(T_S^{(k)})$ and $E(T_S^{(k)})$ as a function of $\mu$ and $d$. In these experiments $k = 1$, $C = 7$, $\Delta = 7$, and $\alpha = \delta$.

$$\begin{aligned} \beta(x,y) &= \mathbb{P}\{C_0 = x, S_0 = y\} \\ &= \binom{C}{x}\mu^x(1-\mu)^{C-x}\binom{s_0}{y}\mu^y(1-\mu)^{s_0-y}. \end{aligned} \tag{3}$$

The second one, that we denote by $\delta$, consists simply in starting from state $(s_0, 0, 0)$, that is the state free from malicious peers, where $s_0 = \lfloor \Delta/2 \rfloor$. We have

$$\delta(s_0, x, y) = 1_{\{x=0, y=0\}}. \tag{4}$$

**Remark** It is important to note that in the remaining of the paper, the notion of "sojourn time" or "steps" refer to the number of events received in the system and not the number of triggered operations (cf. Section 6).

### 6.2.2 Sojourn Time of Protocol$_k$ in Safe States

As described in Section 6 the Markov chain $X$ is reducible, the subset of states $S$ and $P$ are transient and subsets $A_S^m$, $A_S^\ell$ and $A_P^m$ are closed subsets. We start our study by first investigating the distribution of $T_S^{(k)}$ which counts the total time spent by *protocol*$_k$ in the subset of safe states $S$ before absorption. Specifically $T_S^{(k)} = \sum_{n=0}^{\infty} 1_{\{X_n \in S\}}$. Following the results in [24], the expectation of $T_S^{(k)}$ is given by

$$E(T_S^{(k)}) = v(I - R)^{-1}\mathbb{1}. \tag{5}$$

where $v = \alpha_S + \alpha_P(I - M_P)^{-1}M_{PS}$ and $R = M_S + M_{SP}(I - M_P)^{-1}M_{PS}$.

### 6.2.3 Sojourn Time of Protocol$_k$ in Polluted States

In the same way, its expectation is given by

$$E(T_P^{(k)}) = w(I - Q)^{-1}\mathbb{1}, \tag{6}$$

where $w = \alpha_P + \alpha_S(I - M_S)^{-1}M_{SP}$ and $Q = M_P + M_{PS}(I - M_S)^{-1}M_{SP}$.

Figure 4 compares the expected number of operations spent in safe and polluted states before absorption for two protocols, *protocol*$_1$ and *protocol*$_C$, as a function of $\mu$, $d$ and the two initial distributions $\delta$ and $\beta$. In the remaining of the paper we only focus on these two specific protocols. The reason is that *protocol*$_1$ implements the least amount of randomization of the `leave` operation (i.e., $k = 1$) while *protocol*$_C$ implements the largest one (i.e., $k = C$). This will allow us to illustrate the fact that counterintuitively increasing the amount of randomization of the operations does not make them necessarily more resilient to strong adversaries.

The first observation that we can draw from this figure is the impact of the initial distribution on the behavior of the cluster. When this distribution equals to $\delta$ (i.e., the cluster is initially free from malicious peers), the number of operations run in safe states is much larger than the one spent in polluted ones for both protocols. This holds even for very large values of both $\mu$ and $d$. This comes from the combined effects of both the `join` and `leave` operations. The former one prevents new peers to directly belong to the core set, while the latter one randomizes the core set population, demanding accordingly a certain amount of time for the adversary to successfully pollute a cluster. On the other hand, when starting with $\alpha = \beta$, both the core and the spare sets are initially populated




Figure 4: $E(T_S^{(k)})$ (Relation (5)) represented by hatched bars, and $E(T_P^{(k)})$ (Relation (6)) represented by plain bars as a function of $k$, $\mu$ and $d$. In all these experiments $C = 7$, $\Delta = 7$, and for the two figures on the left (resp. right), we have $\alpha = \delta$ (resp. $\alpha = \beta$).

| | $\mu = 0\%$<br>$d \in \{0.0, \ldots, 0.9\}$ | $\mu = 10\%$<br>$d \in \{0.0, \ldots, 0.9\}$ | $\mu = 20\%$<br>$d \in \{0.0, \ldots, 0.9\}$ | $\mu = 30\%$<br>$d \in \{0.0, \ldots, 0.9\}$ |
|---|---|---|---|---|
| $E(T_{S,1}^{(1)})$ | $\{12\}$ | $\{11.975, \ldots, 12.085\}$ | $\{11.857, \ldots, 11.890\}$ | $\{11.646, \ldots, 11.570\}$ |
| $E(T_{S,2}^{(1)})$ | $\{0\}$ | $\{0.008, \ldots, 0.013\}$ | $\{0.035, \ldots, 0.033\}$ | $\{0.067, \ldots, 0.043\}$ |
| $E(T_{P,1}^{(1)})$ | $\{0\}$ | $\{0.012, \ldots, 0.099\}$ | $\{0.085, \ldots, 0.558\}$ | $\{0.246, \ldots, 1.611\}$ |
| $E(T_{P,2}^{(1)})$ | $\{0\}$ | $\{0, \ldots, 0.004\}$ | $\{0.006, \ldots, 0.26\}$ | $\{0.021, \ldots, 0.075\}$ |

Table 2: Successive sojourn time in transient states $S$ and $P$ as a function of $\mu$ and $d$. In these experiments $k = 1$, $C = 7$, $\Delta = 7$, and $\alpha = \delta$.

with malicious peers (proportionally to $\mu$). Hence this requires less effort for the adversary to gain control of the cluster. The second observation is that for a given initial distribution $\alpha$, $protocol_1$ always outperforms $protocol_C$, that is for both a given $\mu$ and a given $d$, $E(T_S^{(1)}) \geq E(T_S^{(C)})$ and $E(T_P^{(1)}) \leq E(T_P^{(C)})$. Finally, another general observation that can be drawn from these graphs is that when $\mu = 0$ we have $E(T_S^{(k)}) + E(T_P^{(k)}) = \lfloor \Delta^2/4 \rfloor = 12$. Actually $\lfloor \Delta^2/4 \rfloor$ corresponds to the maximal expected number of steps before absorption in a one dimensional random walk with borders (here, $\Delta$ is the length between the two borders). Now for increasing values of $d$: for a given $\mu$, $E(T_P^{(k)})$ strictly increases while $E(T_S^{(k)})$ slightly decreases. This is explained by the fact that malicious peers can stay longer in the cluster, increasing accordingly their proportion in the cluster up to pollution. Once polluted, the adversary prevents `split` from occurring (cf. Rule (2)) and thus increases the time spent in polluted states and thus the lifetime of the cluster. Clearly this phenomenon is more noticeable for larger values of $\mu$. Now when $d$ tends to 1, we reach the situation where peers, and in particular malicious peers, can stay forever in their cluster which allow them to quickly gain the quorum in the cluster and by their strategy prevent any safe operation to be triggered even for very small values of $\mu$ (see Table 1). This clearly confirms that churn is a fundamental ingredient to defend against targeted attacks. To summarize, the main lessons learnt from these experiments is that by adequately tuning the value of $d$ ($d$ is a system parameter) according to the resources of the adversary (i.e., the proportion $\mu$ of malicious peers in the system), one can clearly decrease the impact of targeted attacks.

#### 6.2.4 Successive Sojourn Times of Protocol$_k$ in Polluted and Safe States

A deeper investigation of $protocol_k$ can be conducted by studying the duration and frequency of successive sojourn times in subsets $S$ and $P$. For $n \geq 1$, we denote by $T_{S,n}^{(k)}$ (resp. $T_{P,n}^{(k)}$) the distribution of the time spent by the Markov chain $X$ during its $n$-th sojourn in subset $S$ (resp. $P$). Thus the total time spent in subset $S$ (resp. $P$) before reaching subsets $A_S$ and $A_P$ is given by

$$T_S^{(k)} = \sum_{n=1}^{\infty} T_{S,n}^{(k)} \text{ and } T_P^{(k)} = \sum_{n=1}^{\infty} T_{P,n}^{(k)}.$$

From [25], we have for $n \geq 1$ and $\ell \geq 0$

$$E(T_{S,n}^{(k)}) = vG^{n-1}(I - M_S)^{-1}\mathbb{1}, \tag{7}$$

where $v$ is defined in Relation (5) and $G = (I - M_S)^{-1}M_{SP}(I - M_P)^{-1}M_{PS}$, and

$$E(T_{P,n}^{(k)}) = wH^{n-1}(I - M_P)^{-1}\mathbb{1}, \tag{8}$$

where $w$ is defined in Relation 6 and $H = (I - M_P)^{-1}M_{PS}(I - M_S)^{-1}M_{SP}$.

Table 2 shows the expected duration of successive sojourn times in subsets $S$ and $P$ for $protocol_k$, with $k = 1$. The main observation drawn from these experiments is that $E(T_S^{(k)}) \simeq E(T_{S,1}^{(k)})$ and $E(T_P^{(k)}) \simeq E(T_{P,1}^{(k)})$, that is the protocol does not alternate between safe and polluted states. This is very noticeable for small values of $\mu$ while a little bit less for larger ones.

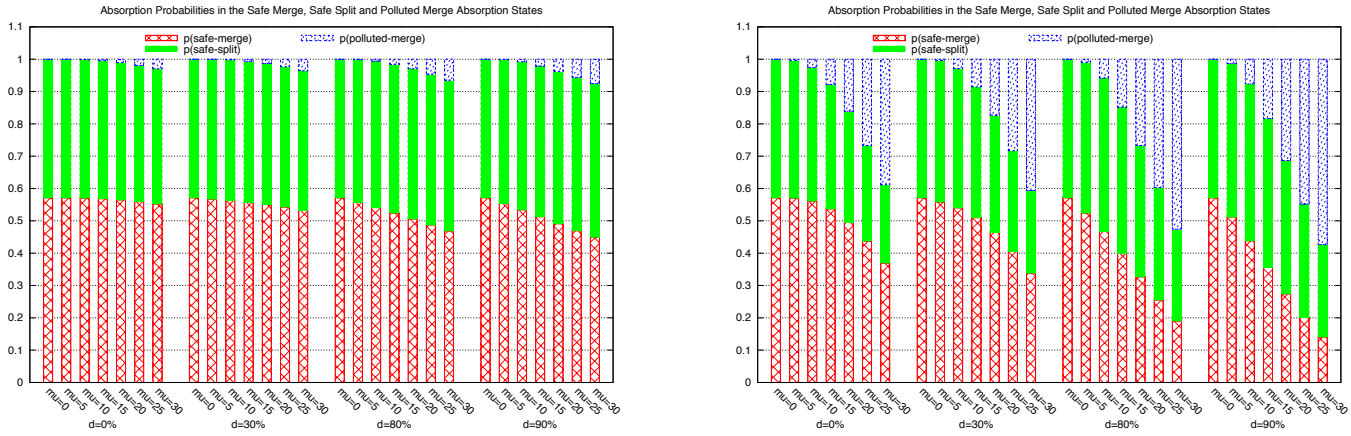



Figure 5: $p(A_S^m)$, $p(A_S^\ell)$ and $p(A_P^m)$ (cf. Relation (9)) respectively represented by red hatched, plain and blue hatched bars as a function of $\mu$ and $d$. In these experiments $k = 1$, $C = 7$, and $\Delta = 7$. We have $\alpha = \delta$ for the left figure and $\alpha = \beta$ for the right one.

### 6.2.5 Absorption Probabilities

Clusters eventually either `split` into two smaller clusters or `merge` with another cluster. An important question to be answered is whether or not `split` operations are more frequent than `merge` ones (for safe clusters), and whether or not polluted clusters `merge` more frequently than safe ones. Rewriting matrix $M$ as

$$M = \begin{pmatrix} T & R_S^m & R_S^\ell & R_P^m \\ 0 & * & * & * \end{pmatrix} \text{ with } T = \begin{pmatrix} M_S & M_{SP} \\ M_{PS} & M_P \end{pmatrix}$$

$$\text{with } R_U^v = \begin{pmatrix} M_{SA_U^v} \\ M_{PA_U^v} \end{pmatrix} \text{ with } U \in \{S, P\} \text{ and } v \in \{m, \ell\},$$

then starting from an initial distribution $\alpha = (\alpha_T \ 0 \ 0 \ 0)$, with $\alpha_T = (\alpha_S \ \alpha_P)$ then the probability $p(A_S^\ell)$ for Markov chain $X$ to be absorbed in states $A_S^\ell$ is

$$p(A_S^\ell) = \alpha_T (I - T)^{-1} R_S^\ell \mathbb{1}. \tag{9}$$

A similar computation gives the probabilities $p(A_S^m)$ and $p(A_P^m)$ to be respectively absorbed in states $A_S^m$ and $A_P^m$.

Figure 5 shows these different probabilities of absorption for $protocol_1$ with the initial distribution $\alpha = \delta$ (on the left graph) and $\alpha = \beta$ (on the right graph). Clearly in absence of adversarial behavior ($\mu = 0$), the cluster remains safe until it `splits` or `merge`, and both operations are equiprobable. Actually, we see that $p(A_S^m) = 0.57$ and $p(A_S^\ell) = 0.43$. This comes from the fact that the initial size $s_0$ of the spare set is equal to $\lfloor \Delta/2 \rfloor = \lfloor 7/2 \rfloor = 3$ (cf. Section 6.2.1), and thus the probability to reach a *merge safe* state equals $1 - 3/7 \simeq 0.57$, and thus a *split safe* state equals 0.43. Thus after some initial period of growth, the topology of the overlay is stable. Let us now observe the influence of the adversary on the probabilities of absorption. For a given $\mu$, the probability for a safe cluster to `split` increases with larger values of $d$. This confirms the results described above: the population of the cluster increases as malicious peers trigger less `leave` operations than `join` operations. However, despite the fact that malicious peers stay longer in the cluster, for $\alpha = \delta$, the probability for the cluster to `merge` in a polluted state is very small (strictly less than 8%) even for very large values of both $\mu$ (e.g., $\mu = 30\%$) and $d$ (e.g., $d = 90\%$). These results are of utmost importance as they show that it is very improbable that polluted clusters manage to contaminate the other clusters of the system. As a consequence this fault-containment feature makes unlikely the probability for a cluster to start in a state in which the population of malicious peers is non negligible, that is from the initial distribution $\beta$. This is confirmed in the next section.

# 7 Modeling the Adversarial Strategy in the Overlay Network

We now analyze the impact of the adversary on the whole overlay, and in particular the long run proportion of polluted clusters. We consider an overlay populated with $n$ clusters $\mathcal{D}_1, \ldots, \mathcal{D}_n$, and subject to join and leave events. Each cluster $\mathcal{D}_i$ implements the same protocol $protocol_k$. 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 $\mathcal{D}_i$ 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 $M$ and the same initial probability distribution $\alpha$. However these chains are not independent since, at each instant, only one Markov chain is allowed to make a transition, this Markov chain being chosen with probability $1/n$. We denote by $N_S^{(n)}(m)$ and $N_P^{(n)}(m)$ 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 $P$ at instant $m$. More formally, these random variables are defined, for $m \geq 0$, by

$$N_S^{(n)}(m) = \sum_{h=1}^{n} 1_{\{X_m^{(h)} \in S\}} \text{ and } N_P^{(n)}(m) = \sum_{h=1}^{n} 1_{\{X_m^{(h)} \in P\}}.$$

It has been proved in [26] that the transient state probabilities of each Markov chain $X^{(h)}$, $h = 1, \ldots, n$ at instant $m \geq 0$ is given by the following theorem.

**Theorem 1** *For every $h = 1, \ldots, n$, $m \geq 0$ and $j \in \Omega$, we have*

$$\mathbb{P}\{X_m^{(h)} = j\} = \sum_{\ell=0}^{m} \binom{m}{\ell} \left(\frac{1}{n}\right)^{\ell} \left(1 - \frac{1}{n}\right)^{m-\ell} \mathbb{P}\{X_\ell = j\} \tag{10}$$

Note that this probability is independent of $h$ since, even though the Markov chains are dependent, they behave identically and each of them is chosen with probability $1/n$.

The expectations of random variables $E(N_S^{(n)}(m))$ and $E(N_P^{(n)}(m))$ are then obtained in the following theorem. We denote by $\mathbb{1}_S$ (resp. $\mathbb{1}_P$) the column vector of dimension $|S \cup P|$ which $i$th entry is equal to 1 (resp. 0) if $i \in S$ and 0 (resp. 1) if $i \in P$.

**Theorem 2** *For every $n \geq 1$ and $m \geq 0$, we have*

$$\frac{E(N_S^{(n)}(m))}{n} = \alpha \left(\frac{1}{n} T + \left(1 - \frac{1}{n}\right) I\right)^m \mathbb{1}_S.$$

$$\frac{E(N_P^{(n)}(m))}{n} = \alpha \left(\frac{1}{n} T + \left(1 - \frac{1}{n}\right) I\right)^m \mathbb{1}_P.$$

**Proof.** From the definition of $N_S^{(n)}(m)$ and using Theorem 1, we have

$$\begin{aligned} E(N_S^{(n)}(m)) &= \sum_{h=1}^{n} \mathbb{P}\{X_m^{(h)} \in S\} = \sum_{h=1}^{n} \sum_{j \in S} \mathbb{P}\{X_m^{(h)} = j\} \\ &= n \sum_{\ell=0}^{m} \binom{m}{\ell} \left(\frac{1}{n}\right)^{\ell} \left(1 - \frac{1}{n}\right)^{m-\ell} \mathbb{P}\{X_\ell \in S\} \end{aligned}$$

It is well-known that the transient state probabilities of generic Markov chain $X$ are given by

$$\mathbb{P}\{X_\ell \in S\} = \alpha T^\ell \mathbb{1}_S.$$

We thus get

$$\begin{aligned} E(N_S^{(n)}(m)) &= n \sum_{\ell=0}^{m} \binom{m}{\ell} \left(\frac{1}{n}\right)^{\ell} \left(1 - \frac{1}{n}\right)^{m-\ell} \alpha T^\ell \mathbb{1}_S \\ &= n\alpha \left(\frac{1}{n} T + \left(1 - \frac{1}{n}\right) I\right)^m \mathbb{1}_S. \end{aligned}$$

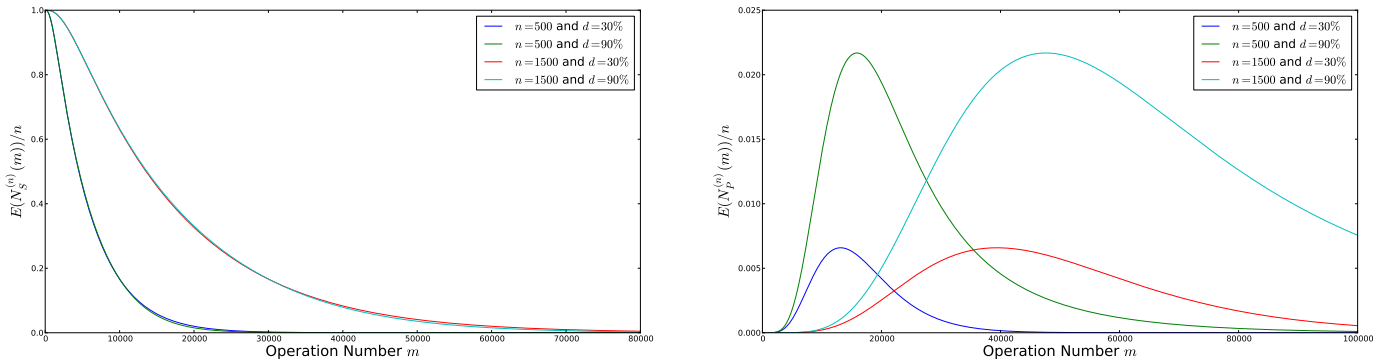



Figure 6: $\frac{E(N_S^{(n)}(m))}{n}$ and $\frac{E(N_P^{(n)}(m))}{n}$ as a function of $m$ for different values of $n$ and $d$.

The expectation $E(N_P^{(n)}(m))$ is obtained using the same lines. ■

The states of $S \cup P$ being transient, matrix $T$ is sub-stochastic and so is matrix $T/n + (1 - 1/n)I$, for every $n \geq 1$. We then have, for every $n \geq 1$,

$$\lim_{m \longrightarrow \infty} \frac{E(N_S^{(n)}(m))}{n} = \lim_{m \longrightarrow \infty} \frac{E(N_P^{(n)}(m))}{n} = 0.$$

Figure 6 depicts the expected proportion of safe (left figure) and polluted (right figure) clusters after the $m^{th}$ transition, i.e., that is the $m^{th}$ `join` or `leave` operation, for realistic values of $n$ and $d$. The main observation is that the expected proportion of polluted (right figure) clusters is very low even for large values of $d$ (less that 2.2%). Let us then observe that the expected proportion of safe clusters is almost independent of $d$ value. The same remark holds for the expected proportion of polluted clusters (even though because of the different y-axis scale used for this figure the phenomenon is not visually straightforward). This independence can be explained by the fact that the real churn dominates the induced churn (represented by the parameter $d$).

# 8 Conclusion

The main lessons learnt from this study is that *(i)* shuffling a single peer at a time ($protocol_1$) performs better than shuffling several peers. This is an interesting property because in this case the implementation is reduced to a single Byzantine tolerant agreement algorithm run amongst spare members, compared to two such runs (an additional one in the core set) for $k > 1$. *(ii)* By choosing an adequate value of $d$ it is possible to noticeably reduce the propagation of attacks while at the same time minimizing the overhead of the induced churn.

# References


[1] N. Naoumov and K. W. Ross, "Exploiting p2p systems for ddos attacks," in *Proceedings of the International Conference on Scalable Information Systems (Infoscale)*, 2006.

[2] A. Fiat, J. Saia, and M. Young, "Making chord robust to byzantine attacks," in *Proceedings of the Annual European Symposium on Algorithms (AESA)*, 2005.

[3] I. Baumgart and S. Mies, "S/kademlia: A practicable approach towards secure key-based routing," in *Proceedings of the International Conference on Parallel and Distributed Systems (ICPADS)*, 2007.

[4] E. Anceaume, F. Brasileiro, R. Ludinard, and A. Ravoaja, "PeerCube: an hypercube-based p2p overlay robust against collusion and churn," in *Proceedings of the International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, 2008.

[5] B. Awerbuch and C. Scheideler, "Towards scalable and robust overay networks," in *Proceedings of the International Workshop on Peer-to-Peer Systems (IPTPS)*, 2007.

[6] P. B. Godfrey, S. Shenker, and I. Stoica, "Minimizing churn in distributed systems," in *Proceedings of the ACM SIGCOMM*, 2006.

[7] E. Anceaume, F. Brasileiro, R. Ludinard, B. Sericola, and F. Tronel, "Analytical study of adversarial strategies in cluster-based overlays," in *Proceedings of the 2nd International Workshop on Reliability, Availability, and Security (WRAS)*, 2009.

[8] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach, "Secure routing for structured peer-to-peer overlay networks," in *Proceedings of the Symposium on Operating Systems Design and Implementation (OSDI)*, 2002.

[9] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in *Proceedings of the ACM SIGCOMM*, 2001.

[10] I. Stoica, D. Liben-Nowell, R. Morris, D. Karger, F. Dabek, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of the ACM SIGCOMM*, 2001.

[11] P. Druschel and A. Rowstron, "Past: A large-scale, persistent peer-to-peer storage utility," in *Proceedings of the Workshop on Hot Topics in Operating Systems (HotOS)*, 2001.

[12] A. Singh, T. Ngan, P. Drushel, and D. Wallach, "Eclipse attacks on overlay networks: Threats and defenses," in *Proceedings of the Conference on Computer Communications (INFOCOM)*, 2006.

[13] E. Sit and R. Morris, "Security considerations for peer-to-peer distributed hash tables," in *Proceedings of the International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[14] K. Hildrum and J. Kubiatowicz, "Asymptotically efficient approaches to fault-tolerance in peer-to-peer networks," in *Proceedings of theInternational Symposium on Distributed Computing (DISC)*, 2003.

[15] T. Condie, V. Kacholia, S. Sankararaman, J. M. Hellerstein, and P. Maniatis, "Induced churn as shelter from routing-table poisoning," in *Procs of the 13th thirteenth Annual Symposium on Network and Distributed System Security (NDSS'06)*, 2006.

[16] B. Awerbuch and C. Scheideler, "Group spreading: A protocol for provably secure distributed name service," in *Proceedings of the 31rst International Colloquium on Automata, Languages and Programming (ICALP)*, 2004.

[17] E. Bortnikov, M. Gurevich, I. Keidar, G. Kliot, and A. Shraer, "Brahms: Byzantine resilient random membership sampling," *Computer Networks*, vol. 53, no. 13, pp. 2340–2359, 2009.

[18] C. G. Plaxton, R. Rajaraman, and A. W. Richa, "Accessing nearby copies of replicated objects in a distributed environment," in *Proceedings of the 9th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, 1997.

[19] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the XOR metric," in *Proceedings for the International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[20] T. Locher, S. Schmid, and R. Wattenhofer, "eQuus: A provably robust and locality-aware peer-to-peer system," in *Proceedings of the International Conference on Peer-to-Peer Computing (P2P)*, 2006.

[21] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, 1982.

[22] J. Douceur, "The sybil attack," in *Proceedings of the International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[23] R. Housley, W. Ford, W. Polk, and D. Solo, "Internet x.509 public key infrastructure certificate and crl profile," 1999.

[24] B. Sericola, "Closed form solution for the distribution of the total time spent in a subset of states of a Markov process during a finite observation period," *Journal of Applied Probability*, vol. 27, no. 3, pp. 713–719, 1990.

[25] B. Sericola and G. Rubino, "Sojourn times in Markov processes," *Journal of Applied Probability*, vol. 26, no. 4, pp. 744–756, 1989.

[26] E. Anceaume, F. Castella, R. Ludinard, and B. Sericola, "Markov chains competing for transitions: Application to large scale distributed systems," INRIA, http://hal.inria.fr/inria-00485667/en/, Tech. Rep., 2010.