LiFT: Lightweight Freerider-Tracking Protocol

Rachid Guerraoui, Kévin Huguenin, Anne-Marie Kermarrec, Maxime Maxime Monod

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

LiFT: Lightweight Freerider-Tracking Protocol

Rachid Guerraoui — Kévin Huguenin — Anne-Marie Kermarrec — Maxime Monod

N° 6913

April 2009

Thème COM

ISSN 0249-6399 ISRN INRIA/RR--6913--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
RENNES - BRETAGNE ATLANTIQUE

LiFT: Lightweight Freerider-Tracking Protocol

Rachid Guerraoui* , Kévin Huguenin , Anne-Marie Kermarrec , Maxime Monod*

Thème COM — Systèmes communicants
Équipe-Projet ASAP

Rapport de recherche n° 6913 — April 2009 — 15 pages

Abstract: This paper addresses the problem of detecting freeriders in peer-to-peer epidemic content dissemination applications, such as gossip-based and mesh-based systems. We present LiFT, a Lightweight Freerider-Tracking Protocol, relying on accountability: each peer logs a digest of its past interactions with other peers and tracks abnormal behaviors by cross-checking its history log with other peers. LiFT is the first protocol that detects freeriders in a randomized push-based application without using cryptography. Such applications are challenging because incentives *à la* Tit-for-Tat are not applicable and the random selection of communication partners prevents the use of any deterministic verification of the logs.

We present a theoretical analysis of LiFT, backed up by extensive simulations. Additionally, we report on our experiments on the illustrative example of a gossip-based content dissemination protocol. In this setting, we show that after 50 gossip periods, LiFT detects freeriders (that freeride the protocol by 10%) over 99% of the time and that only 0.1% of the honest nodes are wrongfully accused.

Key-words: Peer-to-peer systems, Gossip-based protocols, Freeriders, Accountability

* Laboratoire de programmation distribuée (LPD), École Polytechnique Fédérale de Lausanne (EPFL)

Centre de recherche INRIA Rennes – Bretagne Atlantique
IRISA, Campus universitaire de Beaulieu, 35042 Rennes Cedex
Téléphone : +33 2 99 84 71 00 — Télécopie : +33 2 99 84 71 71

LiFT : un protocole léger de détection de pair égoïste

Résumé :

Mots-clés :

1 Introduction

High-bandwidth content dissemination applications are increasingly popular over the Internet. For obvious scalability reasons, the tendency has advantageously shifted to the peer-to-peer paradigm yielding efficient distributed protocols. These include multi tree-based protocols such as [5], mesh-based protocols such as [6, 25, 28], and gossip-based protocols such as [4, 8, 9, 22]. All these distributed dissemination protocols highly rely on the willingness of peers to collaborate, i.e., to devote a fraction of their resources – more specifically their upload bandwidth – to the system.

Not surprisingly, such systems are vulnerable to *freeriders*, i.e., peers that do not contribute their fair share [20]. Freeriding is a common behavior in large-scale systems deployed in the public domain [1]. The impact of freeriders is even more critical when they collude, i.e., they collaborate to increase both their individual and common interest.

Two orthogonal approaches have been considered to cope with freeriding: *incentive* and *coercive* mechanisms. While the first incites peers to share their resources by making their profits proportional to their contribution, the latter punishes peers that do not collaborate. The Tit-for-Tat (TfT) incentive mechanism, where content is exchanged only between peers with mutual interest, is a simple and efficient way to discourage freeriders in a system where dissemination is symmetric, i.e., the exchanges between pairs of peers are balanced and bidirectional [6, 22]. On the other hand, accountability, i.e., logging peers' actions for further verifications using a reference implementation of the protocol, provides a powerful coercive mechanism. So far, such mechanisms have mainly been successfully applied only to *deterministic* protocols [11]. However, none of the proposed solutions is applicable to randomized algorithms such as epidemic protocols without using asymmetric (expensive and not really scalable) cryptography [3, 22].

We believe that a verification mechanism inspired from accountability, acting as a coercive mechanism, can lead *(i)* to a fair system on its own or *(ii)* be a relevant complement to collaboration incentives based on bi-directional exchanges such as TfT. In fact, fear of being detected can itself act as an incentive not to misbehave: considering a non-zero probability of being detected when misbehaving and that detection leads to some punishment (e.g., a penalty or exclusion), nodes might want to cooperate by providing their fair share. Designing a verification mechanism to detect possibly colluding freeriders in a content dissemination system based on random peer selection without cryptography raises however the following challenges: *(i)* how to verify that a peer contributes its fair share when the set of peers it is supposed to upload data to may not be predictable (randomness of peer selection)? ; *(ii)* how to use untrusted logs (no cryptography) in presence of colluding peers that can mutually cover each other up? ; *(iii)* how to distinguish between a message lost by the network and a message dropped on purpose by a freerider?

This paper addresses these issues and presents the first protocol to secure the wide class of epidemic protocols: our work can be used as a complement to existing solutions designed for dissemination protocols based on bidirectional exchanges. Actually, such protocols such as BitTorrent [6] or BAR Gossip [22] include an opportunistic unchocking mechanism, necessary to TfT-based dissemination protocols, that constitutes their Achile's heel in the presence of freeriders [24, 26, 30].

We illustrate our approach through a generic gossip-based protocol, where data is disseminated in a random manner following an *asymmetric* push scheme, as opposed to *symmetric* exchanges where a balance between the interests of a pair of nodes can be achieved by means of TfT. Moreover, gossip-based protocols are increasingly popular and have been successfully applied to streaming [4] and file sharing [8], two of the most attractive Internet applications. Beyond gossip, we believe our approach can be extended to any peer-to-peer content dissemination protocol.

In this setting, we propose LiFT, a Lightweight Freerider-Tracking Protocol, to address the problem of freeriders. LiFT is composed of a set of deterministic and statistical distributed verification procedures. LiFT ensures the contribution of each peer is fair (with respect to the protocol), and is uniformly distributed among the nodes. Interestingly enough, the high dynamic and strong randomness of gossip-based protocols, that may be considered as a difficulty at a first glance, happens to help tracking freeriders: LiFT exploits the fact that nodes pick neighbors at random to alleviate the need for cryptography. The fact that a peer interacts with a large subset of the peers and that they are chosen at random, drastically limits its possibility to freeride without being detected as it prevents it from deterministically choosing colluding partners that would cover it up. LiFT is lightweight and provides a good probability of detecting freeriders while keeping the probability of false positives – i.e., inaccurately classifying a correct node as a freerider – very low. The paper provides a theoretical analysis of LiFT backed up by

extensive simulations. Additionally, we report on our simulated experiments on the illustrative example of a gossip-based content dissemination protocol.

The rest of the paper is organized as follows. Section 2 reviews related work. Our system model is defined in Section 3. Section 4 describes our illustrative gossip protocol and lists the opportunities for freeriders to get the maximum utility out of it without contributing their fair share. Section 5 presents LiFT and Section 6 formally characterizes its performance, which is confirmed by the simulations presented. Section 7 outlines future work in the direction opened by LiFT and concludes the paper.

2 Related Work

The LiFT protocol proposed in this paper aims at preventing freeriding in epidemic content dissemination protocols following a coercive approach. It relates to recent work in various ways we overview below.

Incentives have been broadly used to deal with freeriders in file sharing systems such as BitTorrent [6] by making the benefits of the peers proportional to their contribution to the system. However, there exists several attacks to the popular Tit-for-Tat distributed incentive mechanism allowing selfish peers to download contents without any contribution to the system [24, 26, 30], mainly by taking advantage of peers' generosity, namely the opportunistic unchocking mechanism.

In the particular case of streaming, a novel distributed credit-based incentive scheme rewarding peers for advertising content providers (and not only for their contribution in terms of data upload) has been proposed in [29]. This takes into account the continuous nature of delivered media. Whilst promising and well-suited to streaming applications, this protocol may not be resilient to freeriding as it increases the possibility for self-serving peers to achieve their download without contributing, especially when colluding.

PeerReview [11] deals with malicious nodes following an accountability approach. Peers maintain verifiable signed logs of their actions that can be audited and checked using a simulator of the application, i.e., a reference implementation, run at each peer (in addition to the application). Whilst powerful and highly generic, PeerReview is not applicable to non-deterministic protocols which execution relies on randomized algorithms. A recent work proposed CSAR [3], based on verifiable random functions that allows accountability of randomized algorithms. This is implemented by means of asymmetric cryptography primitives.

Indeed, inherent randomness of epidemic protocols makes the robustification of this class of protocols a hard task. Several solutions have been proposed recently, mainly in the context of streaming applications. Li *et al.* proposed BAR Gossip [22] and its improvement [21], a gossip-based streaming application resilient to freeriders (referred to as rational in the considered BAR model [2]). Randomness is handled similarly to CSAR. BAR Gossip is composed of opportunistic pushes performed by altruistic nodes (essential for the efficiency of the protocol) and balanced pairwise exchanges based on a TfT mechanism made robust using cryptographic primitives. Therefore, it cannot be leveraged to robustify against freeriders a protocol based only on optimistic pushes such as the considered one. In addition, we believe that the use of asymmetric cryptography results in a highly reduced scalability.

In [13], Jelasity *et al.* consider the case of malicious participants in generic gossip protocols, i.e., consisting of state exchanges and updates. They rely on cryptography for signing messages between peers and do not consider malicious behaviors that stem from the partner selection, i.e., biasing the random choices.

The two approaches that relate the most to LiFT are the distributed auditing protocol proposed by Haridasan *et al.* in [12] and the passive monitoring protocol proposed by Karakya *et al.* in [15]. The first protocol targets live streaming applications. Freeriders are detected by cross-checking their neighbors reports. The latter focuses on gossip-based search in the Gnutella network. The peers monitor the way their neighbors forward/answer queries in order to detect freeriders and query droppers. Yet, contrarily to LiFT– which is based on random peer selection – in both protocols, the peers's neighborhoods are static, i.e., forming a fixed mesh overlay. Therefore, the situation where colluding peers mutually cover up (not addressed by the authors) makes such monitoring protocols vain.

The originality of our work lies in exploiting the inherent randomness of epidemic protocols: lightweight information cross-checking and statistical verifications alleviate the need in LiFT for heavy-weight cryptography, even in presence of colluding nodes.

3 System Model

We consider a system of n nodes that communicate over lossy links. We assume that every node can receive incoming data from any other node in the system (because the nodes are either not guarded or firewalled, or there exists a means to circumvent such protections [18,31]) and that delays between nodes are negligible compared to the gossip period T_g. In addition we assume that nodes can pick uniformly at random a set of nodes in the system. This is usually achieved using full membership or a random peer sampling protocol [10,14,19].

Nodes are either honest or freeriders. The first strictly follows the protocol (including verifications for freeriders detection) while the latter allow themselves to deviate from the protocol in order to maximize the profit they get out of it, i.e., the amount of received data while minimizing their contribution to the system, i.e., the amount of sent data, without being blamed. Therefore, we consider neither Denial-of-service like attacks nor pollution [23]. In addition, honest nodes always tell the truth when asked for a testimony during a verification procedure while freeriders may lie not to be detected and thus avoid to be blamed. Yet, freeriders do not wrongfully accuse honest nodes since it does not increase their benefit.[1]

We further consider that freeriders can collude to increase their individual and mutual benefit and possibly cover up each others in case of verification procedures. We denote by m the number of freeriders in the system. However, we assume a sybil-free system and therefore $m \ll n$. Yu *et al.* opened a promising way to deal with sybil attacks in overlay networks [32].

Finally, we assume a central entity, known by each node, that collects complaints about suspected freeriders. More specifically, the central authority stores a score, i.e., a real number, for each node in the system, modified by LiFT by means of *blames*. Note that distributing the score among the overlay is a problem on its own as it implies new possible attacks, e.g., malicious nodes promoting the reputation of colluding nodes. We refer the reader to [27] for further reading on this topic. Similarly, how to punish freeriders, e.g., remove them from the system, is out of the scope of this paper.

4 Freeriding a Gossip Three-phase Protocol

We first describe a basic gossip three-phase protocol, such as the ones successfully used for high-bandwidth content dissemination such as file sharing [8] and streaming [9]. We then exhaustively list the attacks that can be made to the protocol, in each of its phase.

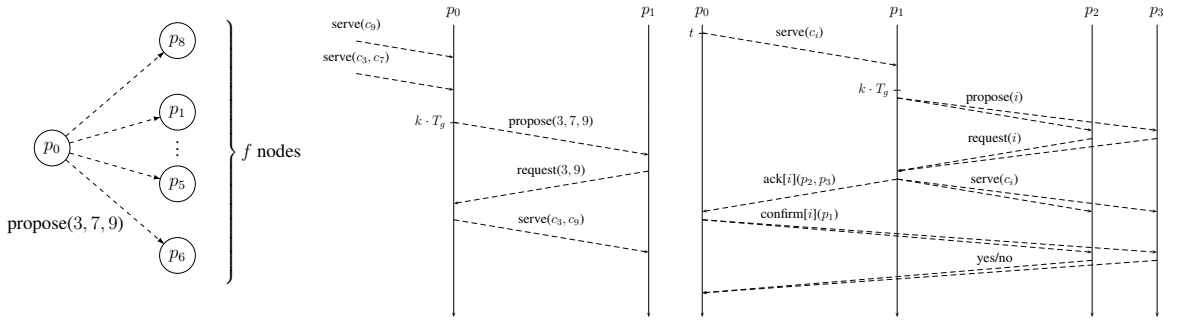
4.1 Basic Gossip Three-phase Protocol

We consider a system where some content is broadcasted from a source to all nodes using a gossip-based protocol. The content is split in multiple chunks that are identified by chunk ids. In short, each node periodically proposes a set of chunks to a set of random nodes (i.e., proposing the chunk ids). Upon reception of a proposal, a node requests back the sender the chunk ids it needs and the sender then pushes the requested chunks. All messages are sent over UDP. The three phases are illustrated in Figures 1a-1b.

Propose phase A node periodically, i.e., every gossip period T_g, picks uniformly at random a set of f nodes and proposes to them (as depicted in Figure 1a) the set $\mathcal{P}$ of chunks it received since its last propose phase. The size f of the node set, namely the *fanout*, is the same for all nodes and kept constant over time. Such gossip protocol follows an *infect-and-die* process as once a node proposed a chunk to a set of nodes, it does not propose it any further.

Request phase Upon reception of a proposal for a set $\mathcal{P}$ of chunks, a node determines the subset of chunks $\mathcal{R}$ it needs and requests them from the sender (even if $\mathcal{R} = \varnothing$). More specifically, the node requests the chunks it has not receive yet.

[1] In the case where nodes detected as freeriders are expelled from the system, falsely accusing honest nodes leads to an increased proportion of freeriders. This phenomenon, known as the *tragedy of the commons* decreases the overall performance of the system, including freeriders.



(a) Gossip (proposals) (b) Three phases per proposal (c) Cross-checking for causality verification

Figure 1: Three-phase gossip and verification protocols

Serving phase When a proposing node receives a request to a proposal, it sends back the corresponding chunks. If the request does not follow a proposal, it is ignored. Similarly, nodes only serve chunks that were proposed (i.e., chunks in $\mathcal{P} \cap \mathcal{R}$).

4.2 Freeriding the Protocol

Freeriders may deviate from the protocol in three ways: *(i)* bias the partner selection, *(ii)* drop messages they are supposed to send, or *(iii)* modify the content of the messages they send. Following this line of reasoning, we exhaustively list all possible attacks in each phase of the protocol, discuss their motivations and impact, and then extract and classify those that may increase the individual interest of a freerider or the common interest of colluding freeriders. In the sequel, attacks that require or serve colluding nodes are denoted with a '$\star$'. This analysis of the three-phase protocol is at the core of our lightweight verification scheme – LiFT.

Propose phase In the first phase, a freerider can *(i)* propose to less than f nodes (possibly not proposing at all), *(ii)* propose less chunks than it should, *(iii)* reduce its proposing rate or *(iv)* select as communication partners only a subset of nodes.

(i) **Propose to less than fanout nodes** By proposing to $\hat{f} < f$ nodes, the freerider trivially reduces the potential number of requests, thus the probability of serving chunks. Therefore, its contribution in terms of the amount of data uploaded is decreased while still fulfilling the $\hat{f}$ received requests.

(ii) **Illegal content** A proposal is valid if it contains every chunks received in the last gossip period, and only those. Proposing only a subset of the chunks received in the last period obviously decreases the number of requested chunks. However, a freerider has no interest in proposing chunks it does not have since that, contrarily to TfT-based protocols uploading chunks to a node does not implies that this latter sends chunks in return. In other words, proposing more (and possibly fake) chunks does not increase a the benefit of a node.

(iii) **Bias the partners selection ($\star$)** Considering a group of colluding nodes, a freerider might want to bias the random selection of partners either *(a)* to privilege its colluding partners, so that the group's benefit increases, or *(b)* to pick nodes that have a lower probability to request chunks, e.g., nodes that already have (or requested) the chunks it is going to propose. This last case requires an accurate global knowledge of the system that is difficult to obtain in a distributed fashion in a large scale system. Therefore, we consider only the first attack.

(iv) **Varying gossip rate** A malicious node may change its gossip period to $T_g' \neq T_g$. Increasing gossip period leads to less frequent proposals, i.e., advertising more chunks per proposal, but "older" ones. This implies a decreased interest for the requesting nodes and thus a decreased contribution for the sender. This is due to the fact that an old chunk has a lower probability to be of interest as it becomes more replicated over time. This is therefore a potential attack for a freerider. The other way around, a malicious node would increase its contribution if decreasing its gossip period.

Pull request phase Nodes are expected to request only chunks they have been proposed. A freerider would increase its benefit by opportunistically requesting extra chunks (even from nodes that did not propose to it). The dissemination protocol itself prevents this misbehaving by automatically dropping such requests as described above. Therefore, no attack in the pull request phase has to be considered.

Serving phase In the serving phase, a node may *(i)* send only a subset of what was requested or *(ii)* send junk. The first obviously decreases a freerider's contribution as the sender serves less chunks than it is supposed to. However, as we mentioned above, in the considered asymmetric epidemic protocol, a freerider has no interest in sending junk data.

Analyzing the three-phase protocol in details allowed to identify possible attacks. Assuming a freerider deviates from the protocol only to increase its individual interest – or the interest of a group of colluding nodes, we reduced the set of potential attacks. Interestingly enough, these attacks share similar aspects and can thus be gathered into three classes that dictate the rationale along which a verification protocol should be designed.

The first is *quantitative correctness* that characterizes the fact that a node effectively proposes to the correct number of nodes (f) at the correct rate ($1/T_g$). Assuming this first aspect is verified, two more aspects must be further considered: *causality* that reflects the correctness of the deterministic part of the protocol, i.e. received chunks must be proposed in the next gossip period as depicted in Figure 1b, and *statistical validity* that evaluates the fairness (with respect to the distributions specified by the protocol) in the random communication partners selection.

5 Lightweight Freerider-Tracking Protocol

In this section we present LiFT, a Lightweight Freerider-Tracking Protocol that encourages nodes, in a coercive way, to contribute their fair share to the system, by means of distributed verifications along the line of accountability.

LiFT consists in a set of *direct* verifications and *a posteriori* auditing procedures. To this aim, every node logs a bounded-size *history* of sent and received messages. More specifically, each node maintains a trace of the events that occurred in the last h seconds, i.e., corresponding to the last $n_h = h/T_g$ gossip periods. During the verification procedures, nodes act either as *auditors* (i.e., verifiers), *suspects* or *witnesses*. An auditor asks an inspected node p_1 (i.e., a suspect) for its history (or a subset of it) and then: *(i)* checks the *statistical validity* of the random choices made by p_1, *(ii)* cross-checks p_1's history with the histories of the nodes involved in a communication with p_1 during the last n_h gossip periods to asses the *quantitative correctness* and the *causality* of p_1's behavior. When a node does not pass a test, it is blamed by a value proportional to the number of invalid pushes, i.e., proposals, requests and serves. Therefore, the blames emitted by the verification procedures are directly comparable and can thus be summed up into a consistent score.

5.1 Direct verifications and cross-checking

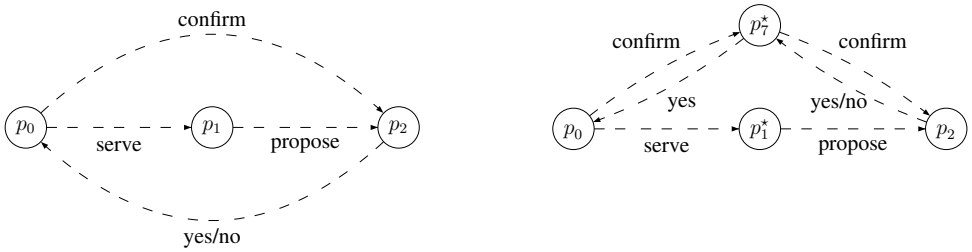
In order to ensure *quantitative correctness*, the verification procedures must verify that the suspected node sent each proposal to f nodes. To assess *causality*, the verification mechanism must ensure that *(i)* a chunk proposed is served, and that *(ii)* a chunk delivered by a node at a time t is proposed in its next propose phase, i.e., at the latest at time $t + T_g$.

It is straightforward for a node to detect that a chunk it has been proposed is effectively served. If not, the requester blames the proposer by $f/|\mathcal{R}|$ (where $\mathcal{R}$ is the set of requested chunks) for each chunk that has not been delivered.

To ensure that received chunks are further proposed, we propose a direct cross-checking verification procedure that works as follows: a node p_1 that received a chunk c_i from p_0 acknowledges to p_0 that it proposed c_i to a set of f nodes as soon as it received their f requests. At this particular moment, p_1 knows that the f nodes received the proposals and thus can confirm that p_1 proposed c_i. At the latest, p_1 sends the list of its f partners T_g seconds after the reception of the chunks: after this length of time, p_1 sends the list regardless of the requests it has received. Figure 1c depicts the message sequence composing a direct cross-checking procedure (with a fanout of 2 for the sake of readability). If anything wrong is

detected, p_1 will blame p_0 by the number of contradictory testimonies or by f if no acknowledgment is sent at all. Direct cross-checking therefore ensures that every node forwards the chunks it receives to exactly f nodes in the next gossip period. Since the verification messages for the direct cross-checking are small and in order to limit the subsequent overhead of LiFT, direct cross-checking is done exclusively with UDP.

Fooling the direct cross-check ($\star$) Considering a set of colluding nodes, a malicious node may freeride the protocol without being detected. Consider the situation depicted in Figure 2a, where p_1 is malicious. If p_0 is malicious and colludes with p_1, then it will not blame p_1, regardless of p_2's answer. Similarly, if p_2 is malicious and colludes with p_1, then it will answer to p_0 that p_1 sent a valid proposal, regardless of what p_1 sent. Even when neither p_0 nor p_2 are malicious or collude with p_1, p_1 can still fool the direct cross-checking thanks to a colluding third party by implementing a *man-in-the-middle attack* as depicted in Figure 2b. Indeed, if a node p_7 is malicious and colludes with p_1, then p_1 can tell p_0 it sent a proposal to p_7 and tell p_2 that the chunk originated from p_7. Doing this, both p_0 and p_2 will not detect that p_1 sent an invalid proposal. The statistical verifications presented in the next section address this issue.



(a) Direct cross-checking (DCC)

(b) Man-in-the-middle attack

Figure 2: Direct cross-checking and attacks. Colluding nodes are denoted with a '$\star$'.

5.2 Local history auditing

As stated in the analysis of the three-phase protocol, for the sake of fairness, the random choices made in the partners selection must be checked. In addition, the example described in the previous paragraph, where freeriders collude to circumvent the verification procedure, highlights the need for statistical verification of a node's contacts, i.e., the nodes that exchanged messages with the audited node, to ensure that it does not collaborate only with a subset of the network, e.g., a set of colluding malicious nodes that would cover up each other. The statistical verification of the history of the nodes, namely *local history auditing*, acts as a complement to the direct cross-checking. The history of a node that biased its partner selection contains a relatively large proportion of colluding nodes. If a small fraction of colluding nodes is present, they will appear more frequently than honest nodes in each other's histories and can therefore be detected. Based on this remark, we propose an entropy-based approach to detect the bias induced by freeriders on the history of nodes and blame them accordingly

Local history auditing aims at verifying that an inspected node p_1 made *legal* choices in the randomized part of the dissemination protocol, with respect to its specification. Effectively, the gossip protocol specifies that a node picks *uniformly at random*, a set of f partners every gossip period T_g. When inspecting the local history of a node, the auditor computes the number of occurrences of each node in the set of proposals emitted by p_1 during the last h seconds. Defining $\mathcal{F}_h$ as the multiset of nodes to whom p_1 sent a proposal during this period (a node may indeed appear more than once in $\mathcal{F}_h$), the distribution $\tilde{d}_h$ of nodes in $\mathcal{F}_h$ characterizes the randomness of the partners selection. We denote by $\tilde{d}_{h,i}$ the number of occurrences of node i ($i \in \{1, \ldots, n\}$) in $\mathcal{F}_h$ normalized by the size of $\mathcal{F}_h$. The quantitative verifications presented in Section 5.1 ensure that a node sent proposals to f nodes at each gossip period and therefore $|\mathcal{F}_h| = n_h \cdot f$. The empirical distribution $\tilde{d}$ is to be compared to the *uniform distribution*.[2]

[2]Note that we do not verify the statistical independence between the successive sets of nodes selected as picking dependent sets of partners cannot serve the interest of a freerider.

The commonly used function to evaluate the similarity between two distributions is the Kullback-Leibler divergence [7]. Measuring the uniformity of the distribution $\tilde{d}$ of p_1's partners boils down to computing its Shannon entropy:

$$H(\tilde{d}_h) = -\sum_i \tilde{d}_{h,i} \log_2(\tilde{d}_{h,i}) \tag{1}$$

The entropy is positive or null and is maximum when every node of the system appears at most once in $\mathcal{F}_h$ (assuming $n_h f < n$). In that case, it is equal to $\log_2(n_h f)$. Evaluating the *uniformity* of the partner selection is achieved by comparing the entropy of the partner distribution to a given threshold γ ($0 \leq \gamma \leq \log_2(n_h f)$) that has to be chosen as a function of the size of the history, the *degree of collusion* allowed and the maximum number of colluding nodes the system must tolerate. On the other hand, since the peer selection service underlying the gossip protocol may not be perfect, the threshold must be tolerant to small deviation with respect to the uniform distribution.

Taking into account these remarks, we now derive a closed-form expression for γ. Consider a freerider that bias partner selection in order to favor colluding freeriders by picking a freerider as partner with probability p_m and an honest node with probability $1 - p_m$. To maximize the entropy of their histories, the freeriders must choose uniformly at random their partners in the chosen class, i.e., honest or colluding freerider. A first requirement to robustify the protocol against colluding attacks is that the number of proposals in a node's history must be greater than the number of freeriders. Otherwise, by proposing chunks only to other freeriders in a round-robin manner, a node may still be able to achieve a maximized entropy. We therefore set h so that $n_h f \lg m$. Given a maximum number of malicious nodes $m^\star$ and a maximum favoring factor $p_m^\star$, the threshold $\gamma^\star$ must be set accordingly to:

$$\gamma^\star = -p_m^\star \log_2\left(\frac{p_m^\star}{m^\star}\right) - (1 - p_m^\star)\log_2\left(\frac{1 - p_m^\star}{n_h f - m^\star}\right) \tag{2}$$

To avoid ill-founded positives caused by the random peer selection, its expected entropy must be greater than the threshold. Otherwise, honest nodes may be wrongfully blamed by the local history auditing mechanism. Numerical values and practical applications are given in Section 6. If its local history does not pass the verification procedure, the inspected node is blamed by $n_h f$.

A local history audit must be coupled with an *a posteriori* cross-checking verification procedure to guarantee the validity of the inspected node history. Cross-checking is achieved by polling all or a subset of the nodes mentioned in the history for an acknowledgment. The inspected node is blamed for each proposal in its history that is not acknowledged by the alleged receiver, exactly as presented in the direct cross-checking verification. Therefore, an inspected freerider replacing colluding nodes by honest nodes in its history in order to pass the entropic check will not be covered by the honest nodes and it will thus be blamed accordingly.

So far we focused on inspecting the multi-set $\mathcal{F}_h$ of nodes to whom a node sent proposals. Due to the possibility man-in-the middle attack presented in Section 5.1, i.e., of using a colluding node to pass the direct verification, a complementary entropic check must be performed on the multi-set of nodes $\mathcal{F}'_h$ that asked the nodes in $\mathcal{F}_h$ for a confirmation, i.e., direct cross-checking. When dealing with an honest node p_0, $\mathcal{F}'_h$ is composed of the nodes that sent proposals (therefore chunks) to p_0 – namely its *fanin*. Therefore, $\mathcal{F}'_h$ shares similar statistical properties with $\mathcal{F}_h$ and can thus be checked similarly, i.e., using an entropic check. On the other hand, when dealing with freeriders that implemented the man-in-the-middle attack, $\mathcal{F}'_h$ contains a large proportion of colluding nodes and thus the freeriders are detected. If the history of the inspected node does not pass the entropic checks (i.e, fanin and fanout), the node is blamed by $n_h f - |\mathcal{F}_h|$.

Local history auditing can be leveraged to check that a node respected the gossip period T_g specified by the protocol. Assuming a correct fanout (thanks to direct cross-checking), checking the gossip period boils down to counting the number of proposals in the local history.

To conclude, in addition to ensuring that statistical properties of the protocol are respected, local history auditing limits the possibility for freeriders to mutually cover up in order to circumvent the verification procedures. The different attacks and corresponding verification procedures are summarized in Table 1.

Local history auditing verifications are sporadically performed by the nodes using TCP connections. The reasons to use TCP are that *(i)* the overhead of establishing a connection is amortized since local

Attack	**Type**	**Detection**
decrease fanout (f)	quantitative	direct cross-check
illegal proposal ($\mathcal{P}$)	causality	direct cross-check
decrease period (T_g)	quantitative	direct cross-check local auditing
bias partners selection ($\star$)	entropy	local auditing *a posteriori* cross-check

Table 1: Summary of potential attacks and associated verifications

history auditing happens sporadically and carries out a large amount of data, i.e., proportional to h, and *(ii)* local auditing is very sensitive to message losses as the potential blame, i.e., $n_h f$, is much larger than for direct verifications.

6 Analysis and evaluation

This section gives a theoretical performance analysis of the blaming mechanism of LiFT. First we analyze its complexity, then we study probabilistically its detection property and false positives in the presence of colluding freeriders and message losses. The present analysis is backed up by Monte-Carlo simulations.

6.1 Complexity

In this section, we evaluate the overhead induced by LiFT on the content dissemination protocol. To this end, we compute the maximum number of verification/blame messages sent by a node during one gossip period. Note that we do not consider statistical verifications in this section as it does not imply a regular overhead but only sporadic message exchanges.

Direct verification Direct verifications do not require any exchange of verification message exchanges as they consist only in comparing the number of chunks requested by the verifier to the number of chunks it really received. However, direct verification may lead to the emission of f blames. The communication overhead caused by direct verifications is therefore $\mathcal{O}(f)$ messages.

Direct cross-checking In order to check that the chunks it sent during the previous gossip-period are further proposed, the verifier must poll the f partners of its f partners. Similarly, a node is polled by f^2 nodes per gossip period on average and must therefore send f^2 replies. Finally, a node must send the list of its current partners to the f nodes (on average) that served chunks to it in the last gossip period. In addition, since a node inspects its f partners, direct cross-checking may lead to the emission of f blames. The communication overhead caused by direct cross-checking is therefore $\mathcal{O}(f^2)$ messages.

The number of messages send by LiFT is $\mathcal{O}(f^2)$, that has to be compared to the number of messages sent by the three-phase protocol itself. That is $f(2 + |\mathcal{R}|)$ – where $\mathcal{R}$ is the set of requested chunks, the two additional messages being the proposal and the request. However, this remains small when compared to the size of the chunks sent by a node, which is several orders of magnitude larger than the verification/blame messages.

6.2 Wrongful blames

This section provides a theoretical analysis of LiFT. This typically can be used to set the parameters to their optimal values. Due to message losses, a node may be wrongfully blamed, i.e., blamed even though it follows the protocol, we call this wrongful blames. Freeriders are additionally blamed for their malicious actions. Therefore, the score distribution among the nodes is expected to be a mixture of two components corresponding respectively to honest nodes and freeriders. In this setting, likelihood maximization algorithms [16] are traditionally used to address decision problems, i.e., to decide whether a node is a freerider or not. Such algorithms are based on the relative score of the nodes and are therefore not sensitive to the wrongful blames. Effectively, wrongful blames have the same impact on honest nodes and freeriders. One may thus think that there is no need to compensate wrongful blames.

However, in the context of content distribution with malicious nodes, two problems may arise: *(i)* malicious nodes are able to decrease the probability to be detected by wrongfully blaming honest nodes (as the decision algorithm is based on relative scores), and *(ii)* the score of a node joining the system is not comparable to those of the nodes already in the system. For these reasons, the impact of wrongful blames on the nodes must be compensated and the decision algorithm must be based on absolute scores.

Considering message losses independently drawn from a Bernoulli distribution of parameter p_l (we denote by p_r the probability of reception: $p_r = 1 - p_l$), we derive a closed-form expression for the expected value of the blames applied to honest nodes during a given timespan. Periodically increasing all scores accordingly leads to an average score of 0 for honest nodes. This way, a fixed threshold can be used to distinguish between honest nodes and freeriders. To this aim, we analyze, for each verification, the situations where message losses cause wrongful blames and evaluate their expected impact.

Direct verification This procedure checks, for each of the f partners a node sent a proposal to, that the proposed and requested chunks are effectively received. For each requested chunk that has not been sent by a node, the node is blamed by $f/|\mathcal{R}|$. If the request is lost, the node is therefore blamed by f (a). Otherwise it is blamed by the proportion of chunks lost (b). The expected blame applied to an honest node (by its f partners), during one gossip period, due to message losses is therefore:

$$\begin{aligned} \tilde{b}_{\text{dv}} &= f \cdot \left[\overbrace{p_r(1-p_r) \cdot f}^{\text{(a)}} + \overbrace{p_r^2 \cdot (1-p_r)\,|\mathcal{R}| \cdot \frac{f}{|\mathcal{R}|}}^{\text{(b)}} \right] \\ \tilde{b}_{\text{dv}} &= p_r(1-p_r^2) \cdot f^2 \end{aligned} \tag{3}$$

Direct cross-checking This procedure checks that each chunk received is proposed to f nodes in the next gossip period. On average, a node receives f proposals during each gossip period. Therefore a node is subject to f direct cross-checking verifications and each verifier asks for a confirmation to the f partners of the inspected node. If not all chunks sent by the verifier in the previous period are received, then the verifier considers the f proposals sent by the receiver as invalid, i.e., incomplete. The inspected node will therefore be blamed by f from this particular verifier (a). On the other hand, for each verifier and for each partner, the inspected node is blamed by 1 if the proposal, the confirmation or the answer to the confirmation is lost (b). The expected blame applied to an honest node (by the f verifiers), during one gossip period, due to message losses is therefore:

$$\begin{aligned} \tilde{b}_{\text{dcc}} &= f \cdot \left[\overbrace{p_r^2(1-p_r^{|R|+1}) \cdot f}^{\text{(a)}} + \overbrace{f \cdot p_r^2 \cdot p_r^{|\mathcal{R}|+1}(1-p_r^3)}^{\text{(b)}} \right] \\ \tilde{b}_{\text{dcc}} &= p_r^2(1-p_r^{|\mathcal{R}|+4}) \cdot f^2 \end{aligned} \tag{4}$$

Figure 3 depicts the distribution of scores in a simulated network of $10,000$ nodes where both direct verifications and cross-checking are performed. The message loss rate p_l has been set to 7% (reflecting the worst case observed on PlanetLab in our preliminary experiments), the fanout f has been set to 12 (as specified in [17], for a $10,000$-node system) and $|\mathcal{R}| = 4$. The scores of the nodes have been increased by 72.945, according to Formulas (3) and (4). We observe that, as expected, the average score is close to zero ($\simeq 0.01$).

Statistical verification This procedure checks the uniformity of the distribution of the nodes in the history of inspected nodes. To this end, we compute the entropy of the distribution and compare its value to a threshold γ, which is a parameter of the system. The distribution of the entropy H of an honest node's history can be estimated by simulations. This allows the system designer to evaluate the probability of wrongful blames when setting the parameter γ. More specifically, for any non-zero positive value $\varepsilon > 0$, we can chose γ such that the probability of wrongfully blaming the inspected node is smaller than ε. Therefore, the expected value of a blame applied to an honest node upon an entropic check is bounded by $\varepsilon \cdot n_h f$. In the following, we assume that γ is set such that $\varepsilon \cdot n_h f$ is negligible compared to other wrongful blames, i.e., $\tilde{b}_{\text{dv}}$ and $\tilde{b}_{\text{dcc}}$.

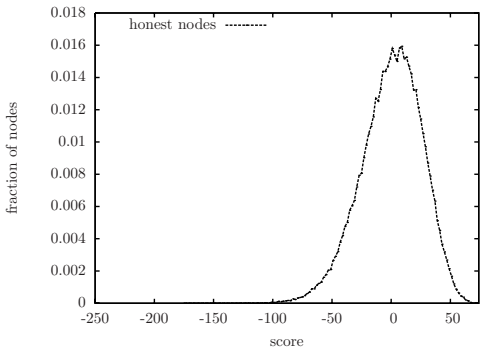
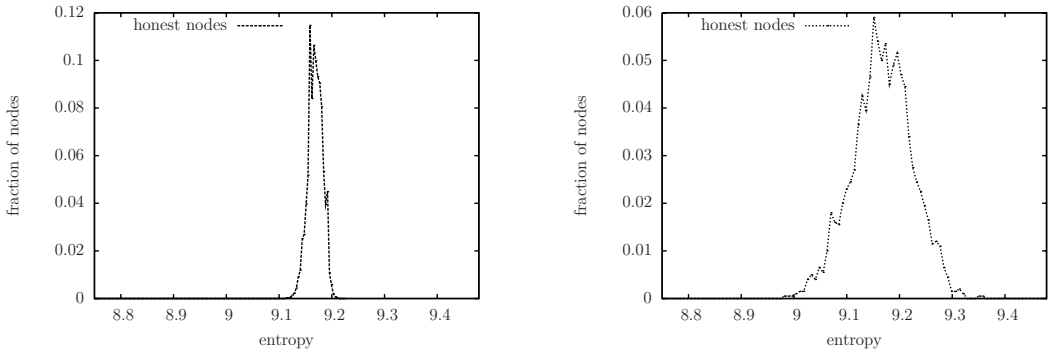


Figure 3: Impact of message losses.

Figure 4a depicts the distribution of entropy H for a history of $n_h f = 600$ partners ($n_h = 50$ and $f = 12$) in a network of $10,000$ nodes. The observed entropy ranges from 9.11 to 9.21 for a maximum reachable value of $\log_2(n_h f) = 9.23$. Similarly, the entropy of the multi-set $\mathcal{F}'_h$, of nodes that selected the inspected node as partner, i.e., its fanin, is depicted in Figure 4b. The observed entropy ranges from 8.98 to 9.34. The threshold γ can therefore be set to $\gamma_0 = 8.95$. Note that the parameter depends on the size of the history $n_h f$. This way, the probability of wrongfully blaming the inspected node is made negligible.



(a) entropy of fanout

(b) entropy of fanin

Figure 4: Distribution of entropy H of the nodes' history for a full membership-based partner selection.

In addition to the entropic check, wrongful blames may be applied to the inspected node in the *a posteriori* cross-check. Effectively, if a proposal sent by the inspected node has not been received by the destination node, due to message losses, this latter will not acknowledge reception when asked. This leads again to wrongful blames. However, since the nodes are polled using TCP, the polling message and the answer are not subject to message losses. On average, only $p_r \cdot n_h f$ proposals in the inspected node history are confirmed by the destination leading to an average blame of:

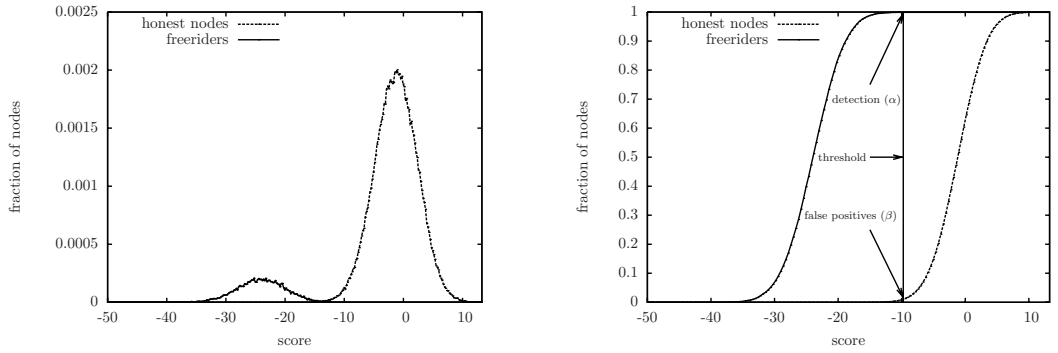
$$\tilde{b}_{\text{apcc}} = (1 - p_r) \cdot n_h f \tag{5}$$

Similarly to direct verification, the wrongful blames applied by the local auditing must be compensated. However, this should be done only sporadically, i.e., only when a node is effectively audited, since these verifications are not triggered at each gossip period.

6.3 Detection

We now evaluate the ability of LiFT to detect freeriders. As mentioned above, the detection mechanism based on the scores of the node must use a fixed threshold. The score of each node is normalized by the number of gossip periods the node spent in the system and compared to this threshold. Any node with a normalized score beyond the threshold is considered as a freerider and punished accordingly. The

theoretical analysis of LiFT presented in the previous section enables the system designer to predict, for any value of the threshold, the probability that an honest node be considered as a freerider, i.e., the probability β of false positives. Obviously, the probability α to catch a freerider depends on its *degree of freeriding*. More specifically, a freerider that serves only one third of the proposed chunk will be blamed more than a freerider serving one half of the chunks. Therefore, the first is more likely to be detected. Formally, we define the degree of freeriding as follows: a freerider of degree δ ($0 < \delta < 1$) contacts only $\delta \cdot f$ nodes per gossip period, proposes a proportion δ of the chunks it received, and serves $\delta \cdot |\mathcal{R}|$ chunks to each requesting node. The benefit of a freerider in terms of the upload bandwidth saved is a function of δ. Note that following the same line of reasoning as in the previous section, we can derive closed form expressions that predict the expected blame applied to a freerider as a function of δ. Due to space limitations, the rest of the presentation focuses on concrete numerical applications obtained by simulated experiments.



(a) probability density function (pdf)

(b) cumulative density function (cdf)

Figure 5: Distribution of normalized scores in the presence of freeriders ($\delta = 0.1$).

Figure 5 depicts the distribution of normalized scores in the presence of 1000 freeriders of degree $\delta = 0.1$ in a 10,000-node system after 50 gossip periods. We plot separately the distribution of scores among honest nodes and freeriders. As expected, the probability density function (Figure 5a) is split into two disjoint modes separated by a gap: the lowest (i.e., left most) mode corresponds to freeriders and the highest one to honest peers. Figure 5b depicts the cumulative density function of scores and illustrates the notion of detection and false positives for a given value of the detection threshold (i.e., -9.75).

Note that since the standard deviation σ of the distribution of normalized scores decreases with the number of gossip periods spent in the system (i.e., $\sigma = \mathcal{O}(1/\sqrt{r})$ where r is the lifetime of the nodes), the performance of LiFT increases over time. Effectively, as the detection threshold is fixed and the distribution is narrower over time the probability of detection α increases and the probability of false positive β decreases.

We set the detection threshold to -9.75 so that the probability of false positive is lower than 1%, and we observe the proportion of freeriders detected by LiFT for several values of the degree of freeriding. Figure 6 represents the proportion of freeriders detected as a function of their degree of freeriding. For instance, one can see that for a node that freerides by 5%, the probability to be detected by LiFT is 65%. Beyond 10% of freeriding, a node is detected over 99% of the time. It is commonly assumed that users are willing to use a modified version of the client application only if it increases signicantly their benefit (resp. decreases their contribution). At the core of FlightPath lies the assumption that the threshold is arround 10%, that matches the performance of LiFT (Figure 6).

As we mentioned earlier, a node can freeride without being detected with the help of colluding freeriders. Such behaviors are possible only to a limited extend determined by the threshold γ, thanks to statistical verifications. In order to prevent from wrongful blames with high probability in the entropic check, in the settings of our experiments, we showed that the threshold must be set to $\gamma_0 = 8.5$. Inverting Formula (2), we deduce that a freerider colluding with 25 other nodes can serve its colluding partners 21% of the time, without being detected. In this setting, a freerider can therefore further decrease its contribution by 21%.

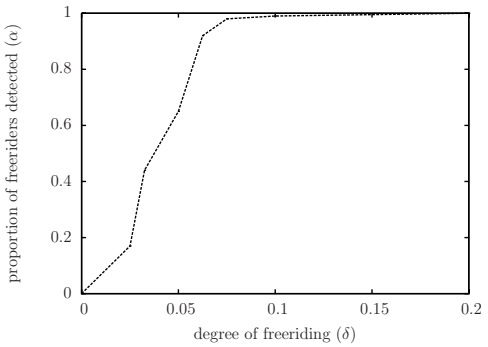


Figure 6: Proportion of freeriders detected by LiFT

7 Conclusion

We presented LiFT, a Lightweight Freerider-Tracking Protocol illustrated as a companion protocol to an epidemic high-bandwidth content dissemination protocol. Beyond the fact that LiFT deals with the inherent randomness of the protocol, LiFT precisely relies on this randomness to robustify its verification mechanisms against colluding freeriders without cryptography. We provided a first order (i.e., based on average expected values of blame and entropy) analysis of LiFT that allows system designers to set the parameters of LiFT to their optimal value and conducted simulated experimentations to evaluate its performance.

The simulation results as well as preliminary ongoing work are very promising, and therefore encourage us to further study the coercive approach at the core of LiFT. We plan to investigate the following tracks. First, we want to extensively test LiFT on top of a real streaming protocol deployed on PlanetLab in order to evaluate the overhead of LiFT in terms of bandwidth consumption, memory and CPU usage. Second, we would like to further analyze the trade-offs between the different parameters of the protocol. For instance finding an optimal balance between the frequency of sporadic *a posteriori* verifications on the one hand, and the overhead and the detection performance of LiFT on the other hand. Finally, detecting colluding freeriders based on time-varying communication graph analysis is a relevant complement to LiFT we would like to work on.

References

[1] E. Adar and B. Huberman. Free riding on Gnutella. *First Monday*, 5(10), October 2000.

[2] A. Aiyer, L. Alvisi, A. Clement, M. Dahlin, J.-P. Martin, and C. Porth. BAR Fault Tolerance for Cooperative Services. *OSR*, 39(5):45–58, 2005.

[3] M. Backes, P. Druschel, A. Haeberlen, and D. Unruh. CSAR: A Practical and Provable Technique to Make Randomized Systems Accountable. In *NDSS*, 2009.

[4] T. Bonald, L. Massoulié, F. Mathieu, D. Perino, and A. Twigg. Epidemic Live Streaming: Optimal Performance Trade-offs. In *SIGMETRICS*, 2008.

[5] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: High-bandwidth Multicast in Cooperative Environments. In *SOSP*, 2003.

[6] B. Cohen. Incentives Build Robustness in BitTorrent. In *P2P Econ*, 2003.

[7] T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley and Sons, Inc., 1991.

[8] M. Deshpande, B. Xing, I. Lazardis, B. Hore, N. Venkatasubramanian, and S. Mehrotra. CREW: A Gossip-based Flash-Dissemination System. In *ICDCS*, 2006.

[9] D. Frey, R. Guerraoui, A.-M. Kermarrec, M. Monod, and V. Quéma. Stretching Gossip with Live Streaming. In *DSN*, 2009.

[10] C. Gkantsidis, M. Mihail, and A. Saberi. Random Walks in Peer-to-peer Networks. In *INFOCOM*, 2004.

[11] A. Haeberlen, P. Kouznetsov, and P. Druschel. PeerReview: Practical Accountability for Distributed Systems. In *SOSP*, 2007.

[12] M. Haridasan, I. Jansch-Porto, and R. Van Renesse. Enforcing Fairness in a Live-Streaming System. In *MMCN*, 2008.

[13] M. Jelasity, A. Montresor, and O. Babaoglu. Detection and Removal of Malicious Peers in Gossip-Based Protocols. In *FuDiCo*, 2004.

[14] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. Gossip-based Peer Sampling. *TOCS*, 25(3):1–36, 2007.

[15] M. Karakaya, I. Körpeoğlu, and O. Ulusoy. Counteracting Free-riding in Peer-to-Peer Networks. *Computer Networks*, 52(3):675–694, 2008.

[16] S. Kay. *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice Hall, 1993.

[17] A.-M. Kermarrec, L. Massoulié, and A. J. Ganesh. Probabilistic Reliable Dissemination in Large-Scale Systems. *TPDS*, 14(3):248–258, 2003.

[18] A.-M. Kermarrec, A. Pace, V. Quéma, and V. Schiavoni. NAT-resilient Gossip Peer Sampling. In *ICDCS*, 2009.

[19] V. King and J. Saia. Choosing a Random Peer. In *PODC*, 2004.

[20] R. Krishnan, M. Smith, Z. Tang, and R. Telang. The Impact of Free-Riding on Peer-to-Peer Networks. In *HICSS*, 2004.

[21] H. Li, A. Clement, M. Marchetti, M. Kapritsos, L. Robinson, L. Alvisi, and M. Dahlin. FlightPath: Obedience v.s. Choice in Cooperative Services. In *OSDI*, 2008.

[22] H. Li, A. Clement, E. Wong, J. Napper, I. Roy, L. Alvisi, and M. Dahlin. BAR Gossip. In *OSDI*, 2006.

[23] J. Liang, R. Kumar, Y. Xi, and K. Ross. Pollution in P2P File Sharing Systems. In *INFOCOM*, 2005.

[24] T. Locher, P. Moor, S. Schmid, and R. Wattenhofer. Free Riding in BitTorrent is Cheap. In *HotNets*, 2006.

[25] N. Magharei, R. Rejaie, and Y. Guo. Mesh or Multiple-Tree: A Comparative Study of Live P2P Streaming Approaches. In *INFOCOM*, 2007.

[26] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani. Do Incentives Build Robustness in BitTorrent? In *NSDI*, 2007.

[27] M. Piatek, T. Isdal, A. Krishnamurthy, and T. Anderson. One Hop Reputations for Peer-to-Peer File Sharing Workloads. In *NSDI*, 2008.

[28] F. Picconi and L. Massoulié. Is There a Future for Mesh-based Live Video Streaming? In *P2P*, 2008.

[29] T. Silverston, O. Fourmaux, and J. Crowcroft. Towards an Incentive Mechanism for Peer-to-Peer Multimedia Live Streaming Systems. In *P2P*, 2008.

[30] M. Sirivianos, J. Park, R. Chen, and X. Yang. Free-riding in BitTorrent with the Large View Exploit. In *IPTPS*, 2007.

[31] W. Wang, C. Jin, and S. Jamin. Network Overlay Construction Under Limited End-to-End Reachability. In *INFOCOM*, pages 2124–2134, 2005.

[32] H. Yu, P. Gibbons, M. Kaminsky, and F. Xiao. SybilLimit: A Near-Optimal Social Network Defense against Sybil Attacks. In *SP*, 2008.

Centre de recherche INRIA Rennes – Bretagne Atlantique
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399