UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

FLÁVIO ROBERTO SANTOS

Slowing Down to Speed Up: Protecting Users Against Massive Attacks in Content Distribution Systems

Thesis presented in partial fulfillment
of the requirements for the degree of
Doctor of Computer Science

Prof. Ph.D. Luciano Paschoal Gaspary
Advisor

Prof. Ph.D. Marinho Pilla Barcellos
Coadvisor

Porto Alegre, July 2013

CIP – CATALOGING-IN-PUBLICATION

Santos, Flávio Roberto

Slowing Down to Speed Up:
Protecting Users Against Massive Attacks in Content Distribution Systems / Flávio Roberto Santos. – Porto Alegre: PPGC da UFRGS, 2013.

103 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2013. Advisor: Luciano Paschoal Gaspary; Coadvisor: Marinho Pilla Barcellos.

1. Content distribution systems. 2. Peer-to-peer systems. 3. File sharing. 4. Streaming systems. 5. Tagging systems. 6. Conservative strategies. 7. Delaying mechanisms. 8. Content pollution. 9. Massive attacks. I. Gaspary, Luciano Paschoal. II. Barcellos, Marinho Pilla. III. Título.

“The difference between a successful person and others is not a lack of strength, not a lack of knowledge, but a lack of will.”

— VINCENT THOMAS LOMBARDI

AGRADECIMENTOS / *ACKNOWLEDGMENTS*

Primeiramente gostaria de agradecer aos meus pais por me permitirem chegar onde cheguei. Em especial agradeço à minha mãe, que é a principal responsável por todas as minhas conquistas. Obrigado aos meus irmãos, Fábio e Felipe, e a todos os familiares que de alguma forma me deram um empurrãozinho. Muito obrigado também aos integrantes do Grupo de Redes da Universidade Federal do Rio Grande do Sul, com destaque ao meu irmão/parceiro Weverton (Tchê Pará), pelas incontáveis discussões, bebedeiras e viagens. A caminhada teria sido muito mais difícil sem todo o companheirismo e cumplicidade de todos. Obrigado aos meus orientadores, Luciano Gaspary e Marinho Barcellos, por terem aceitado o desafio de orientar uma pessoa tão cabeça dura como eu. Obrigado pela excelência e dedicação sempre presentes na condução dos trabalhos. Registro também minha gratidão aos professores e ao corpo técnico-administrativo do Instituto de Informática pelo pronto-atendimento e a seriedade com a qual conduzem seus trabalhos.

Obrigado à galera da pensão do Eraldo e da Housing 158/164, repúblicas onde morei ao longo dos quase quatro anos que passei em Porto Alegre. Aprendi bastante com tanta gente tão diferente juntas. Diversas conversas agradáveis e momentos de companheirismo tornaram essa estadia inesquecível. Muito obrigado a todos os amigos cultivados em terras gaúchas. Agradeço também a compreensão e paciência da minha namorada, Camila, que suportou a distância durante grande parte do meu doutorado. Nesse contexto, o plano Infinity da TIM, a falecida Webjet e o Skype tiveram sua importante contribuição (risos).

There are also some people abroad that I would like to mention. Special thanks to my advisor Burkhard Stiller, who gave me the opportunity to spend one year as a guest researcher in the Communication Systems Group at the University of Zurich, Switzerland. I would like also to thank all my friends and colleagues in Switzerland for their valuable discussions and great time there. Vielen Dank!

Muito obrigado a todos aqueles que contribuiram para minha formação de pesquisador, o que inclui também os amigos do Laboratório de Sistemas Distribuídos da Universidade Federal de Campina Grande, onde fui graduado. Por fim, espero um dia poder retribuir de alguma forma a todo o povo brasileiro, que mesmo sem saber, financiou grande parte dos meus estudos.

CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS

CCDF	Complementary Cumulative Distribution Function
CDF	Cumulative Distribution Function
CDS	Content Distribution Systems
DES	Discrete-Event Simulation
DHT	Distributed Hash Table
DoS	Denial-of-Service
HTTP	Hypertext Transfer Protocol
ISP	Internet Service Provider
LRF	Local Rarest First
MP3	MPEG-1 or MPEG-2 Audio Layer III
NAT	Network Address Translation
P2P	Peer-to-Peer
PEX	Peer Exchange
QoE	Quality of Experience
TFT	Tit-For-Tat

LIST OF SYMBOLS

R	Binary reputation calculated using positive and negative votes.
p	Number of positive votes issued by users.
n	Number of negative votes issued by users.
r	Threshold for R to stop controlling concurrent downloads.
A_{min}	Number of allowed concurrent downloads when $R = 0$.
A_{max}	Number of allowed concurrent downloads when $R \rightarrow 1$.
A	Number of allowed concurrent downloads calculated in terms of R.
D	Number of concurrent downloads taking place in the system.
I	Initial number of seeders uploading a content when it is published.
C	Number of honest users who join the system to download a content.
M	Proportion of malicious users who arrive to attack the system.
Δ	Vocabulary variation calculated using the relative frequencies of tags.
Φ_t	Relative frequencies of a tag t.
σ_t	Standard deviation for the relative frequencies of a tag t.
δ	Threshold for Δ to stop controlling concurrent downloads.
λ	Arrival rate function which determines users' behaviors.
$\overline{\lambda}$	Rate at which users are allowed to join the system.
$\mathcal{Q}$	Quality of experience metric used to evaluate the delaying strategy.
$\mathcal{W}$	Average waiting time used to measure the overhead on users.

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The Internet has become a large platform where users can interact and share personal files or third-party productions. Considering the increasing demand for efficient content sharing, modern and robust *content distribution systems* (CDS) need to be deployed and maintained. In the context of this thesis, CDS are defined as systems used for sharing any kind of content on the Internet. Two categories of CDS are underscored as the most popular ones: file sharing and streaming systems.

Peer-to-peer (P2P) architectures have emerged as a potential solution to improve content dissemination in CDS. The popularization of P2P architectures, in the context of CDS, motivated the scientific community to investigate some challenging problems, namely network topology optimization, bootstrap mechanisms, and service discovery. One particular interesting challenge, in the context of this thesis, is related to mechanisms to approximate users to their personal interests. This is important to guarantee good *quality of experience* (QoE) to users when searching for content. Imprecise descriptions are likely to happen due to different users' opinion or malicious behavior.

Substantial research has been carried out to fight content pollution in CDS. Proposed approaches try to identify and isolate suspicious content after publication. The rationale is to build a base of knowledge about polluters and fake content. However, the reaction time until a content is considered polluted is considerably long, which allows pollution to get widely disseminated. Furthermore, some previous approaches attempt to polarize contents in either polluted or not, not taking into account the inherent *subjectivity* behind the evaluation of shared contents.

The main objective of this thesis is to devise a mechanism to provide users a good QoE – by acting proactively in the early stages of content distribution life cycle – and reduce the effect of malicious interferences. To achieve that, three main steps guided the research work presented in this thesis. First, we proposed a novel strategy that operates conservatively to avoid wide pollution dissemination. Second, we extended our previous solution to cope with the subjectivity regarding content descriptions. Third, and last, we address the pollution attack as a massive attack. To evaluate our solution, a set of experiments was carried out using both real tests and simulations. Results showed the importance of adopting security measures to mitigate malicious behavior in CDS. In the absence of countermeasure mechanisms, even a small proportion (10%) of attackers was able to subvert the system. The introduction of a conservative strategy in this thesis demonstrated the efficacy of delaying users in circumventing massive attacks.

Keywords: Content distribution systems, peer-to-peer systems, file sharing, streaming systems, tagging systems, conservative strategies, delaying mechanisms, content pollution, massive attacks.

Atrasar para Aprimorar: Protegendo Usuários Contra Ataques Massivos em Sistemas de Distribuição de Conteúdo

RESUMO

A Internet tem se tornado uma plataforma importante para interação e compartilhamento de arquivos, o que motivou uma crescente demanda por serviços eficientes. *Sistemas de distribuição de conteúdo* (CDS) precisaram ser criados visando modernidade e robustez. No contexto desta tese, CDS são definidos como sistemas usados para compartilhar qualquer tipo de conteúdo na Internet. Duas categorias de CDS se destacam como as mais populares: compartilhamento de arquivos e sistemas de mídia contínua.

Arquiteturas par-a-par (P2P) surgiram como potenciais soluções para o aprimoramento da disseminação de conteúdo nos CDS. Nesse contexto, a popularização das arquiteturas P2P motivou a comunidade científica a investigar alguns aspectos de pesquisa desafiadores, e.g., otimização de topologias de redes, mecanismos de inicialização de sistemas e serviços de descoberta de recursos. Um desafio com interesse especial a esta tese diz respeito a mecanismos para conciliar a preferência dos usuários aos conteúdos publicados. Esse aspecto é importante para garantir uma boa *qualidade de experiência* (QoE) aos usuários dos sistemas, uma vez que podem existir divergências entre opiniões na descrição dos conteúdos e ações maliciosas.

Esforços de pesquisa constantes têm sido feitos para combater poluição de conteúdo em CDS. Abordagens buscam construir uma base de conhecimento sobre poluidores e conteúdos poluídos para identificar e isolar conteúdos suspeitos depois que eles são publicados. Entretanto, o tempo de reação dessas abordagens até considerar um conteúdo poluído é consideravelmente longo, permitindo uma ampla disseminação de poluição. Além disso, algumas abordagens anteriores buscam polarizar conteúdos entre poluídos ou não, desconsiderando a intrínseca *subjetividade* acerca da classificação dos conteúdos compartilhados.

O objetivo principal desta tese é propor um mecanismo para prover uma boa QoE aos usuários – agindo proativamente durante as fases iniciais da publicação dos conteúdos – e reduzir os efeitos de interferências maliciosas. Para alcançar tal objetivo, três passos principais guiaram o trabalho de pesquisa apresentado nesta tese. Primeiro, propusemos uma estratégia inovadora que opera de forma conservadora para conter a disseminação de poluição. Segundo, estendemos nossa solução para lidar com a subjetividade acerca das descrições dos conteúdos. Terceiro, tratamos o ataque de poluição como um ataque massivo. Para avaliar a solução, experimentos foram executados utilizando testes reais e simulações. Resultados ressaltaram a importância de adotar medidas de segurança para combater comportamentos maliciosos em CDS. Na ausência de mecanismos de contramedida, pequenas proporções (10%) de atacantes foram capazes de comprometer o sistema. A instanciação da estratégia conservadora proposta nesta tese demonstrou a eficácia em atrasar usuários para contornar ataques massivos.

Palavras-chave: Sistemas de distribuição de conteúdo, sistemas par-a-par, compartilhamento de arquivos, sistemas de mídia contínua, sistemas de anotações, estratégias conservadoras, mecanismos de atraso, poluição de conteúdo, ataques massivos.

1 INTRODUCTION

The Internet has become a large platform where users can interact and share contents. These contents are typically personal files or third-party productions, which are published and downloaded by users based on their own interests (ANDRADE et al., 2005). Recent measurement reports have indicated that there is a noticeable number of content sharing activity taking place on the Internet, and that the exchange of multimedia contents accounts for an expressive traffic volume on Internet backbones (GERBER; DOVERSPIKE, 2011). For example, the content uploaded to YouTube in 60 days is equivalent to the amount that would have been broadcast for 60 years, without interruption, by NBC, CBS and ABC altogether (The New York Times, 2010; FIGUEIREDO; BENEVENUTO; ALMEIDA, 2011). To satisfy the demand for efficient content sharing, modern and robust *content distribution systems* (CDS) need to be deployed and maintained (AGER et al., 2011).

In the context of this thesis, CDS are defined as systems used for sharing any kind of content on the Internet. Two categories of CDS are underscored as the most popular ones: file sharing and streaming systems. File sharing systems are used to store and disseminate any digital content, from documents and pictures to audios and videos. Since files are accessed only after the entire download is complete, there is no restriction regarding the order in which bytes are obtained. In contrast, streaming systems are used to disseminate media (e.g., audio and/or video) in a timely manner in order to enable its prompt reproduction. If the content is being broadcast while recorded, the streaming system is classified as "live". Live streaming systems demand efficient network connections to disseminate high-quality streams on the Internet.

Peer-to-peer (P2P) architectures have emerged as an alternative to speed up content dissemination in CDS (COHEN, 2003; ZHANG; LIU; PETER YUM, 2005; LIAO et al., 2006; HECHT et al., 2011). These architectures help balancing out the server bandwidth load. As a result, overload is removed from the network near the server, and the bandwidth requirements are spread over the network of users (peers). Instead of maintaining central entities to distribute content, P2P systems are designed so as peers can interact to share data and maximize their download bandwidth usage. The adoption of P2P architectures, in the context of CDS, has attracted attention from the scientific community to some interesting challenges, namely network topology optimization (CHOFFNES; BUSTAMANTE, 2008; CHOFFNES; SÁNCHEZ; BUSTAMANTE, 2010; POESE et al., 2010; CAPOTA et al., 2011), bootstrap mechanisms (TADDIA; MAZZINI, 2008), and service discovery (IAMNITCHI; FOSTER, 2001). One particular interesting challenge, in the context of this thesis, refers to mechanisms employed to match user's interests and published contents.

The efficacy of CDS depends on the expertise of publishers to properly describe con-

tents, which comprises an important task to guarantee good *quality of experience* (QoE) to users. Since users typically have different opinions, similar contents may be classified with unrelated descriptions. Moreover, due to malicious behavior, attackers can publish contents with popular descriptions to deviate users' attention, obfuscate their queries, and/or promote marketing campaigns. Concrete evidences and statistics about "pollution[1]" have been widely reported in the recent past (e.g., in BitTorrent communities (TORRENTFREAK, 2009; CUEVAS et al., 2010) and KaZaa network (LIANG et al., 2005)). These evidences are underscored by a previous study (SANTOS et al., 2011) and the results of a survey we have carried out with administrators of BitTorrent communities, who informed that around 25% of contents being published are "polluted" and need manual intervention.

There has been substantial research on means to mitigate content pollution in CDS, leading to the proposal of several generic countermeasure mechanisms (CAI et al., 2009; ANAND; BHASKAR, 2011; SHIN; REEVES, 2012). Although these are interesting strategies to fight pollution, they try to identify and isolate suspicious content after publication. However, attackers can easily generate multiple fake content and keep inserting malicious decoys. To circumvent that, alternative and complementary solutions focused on detecting, in addition to fake content, the users responsible for uploading them. One limitation of these solutions is that the time required to detect these copies and/or polluters may be long enough to allow the wide pollution dissemination (*a*). Another limitation is the fact that previous approaches attempt to polarize contents in either polluted or not (*b*). Therefore, they do not take into account the inherent *subjectivity* behind the evaluation of shared contents.

The main objective of this thesis is to devise a mechanism to provide users a good QoE – by acting proactively in the early stages of content distribution life cycle – and reduce the effect of malicious interferences. To achieve that, the research work presented in this thesis was carried out in three main steps. In the first iteration towards a solution to protect users from malicious behaviors, we focused on the limitation (*a*) and proposed a novel strategy that operates conservatively to avoid wide dissemination of polluted content in early stages. The strategy, called FUNNEL, defines a reputation for contents based on binary votes (positive and negative) and regulates the access to potential fake contents. As content reputation increases, more users are allowed to download it. Otherwise, if the reputation decreases, fewer downloads are authorized. Although FUNNEL behaves proactively to protect systems from suspicious polluted content, it requires some sort of expertise from users to assign precise votes.

It is important to notice that contents regarded as "polluted" by a certain user might be considered "non-polluted" by another. Furthermore, Lee et al. (LEE et al., 2006) have shown that about 70% of users fail to notice certain types of incorrect descriptions. Such a fact is neglected in the design of existing approaches, which assume that non-malicious users always detect inappropriate metadata. In the second iteration, we advanced our previous solution to cope with limitation (*b*). The binary classification was replaced by a free vocabulary, which allows users to describe contents using annotations (tags). This novel approach, called DÉGRADÉ, improves user's expectations by moving beyond content polarization (polluted or not) and providing a flexible platform to describe contents.

In the third and last iteration, we generalized the pollution attack to a massive attack, in which malicious users arrive *en masse* to harm the overall system. We proposed a

[1]In the context of this thesis, *content pollution* represents the act of intentionally uploading mislabeled content or later describing them with imprecise metadata.

novel analytic model to represent users' arrival and impose artificial delays to diminish massive attacks. The effect of delaying users "dilutes" malicious activities and gives honest users the chance to join before potential attackers. Different delaying strategies may be applied to maximize the overall QoE depending on specificities of the targeted system. In conclusion, the incremental path taken in this thesis allowed us to abstract the rationale behind the conservative strategy to formulate our hypothesis that **"delaying user's actions mitigates massive attacks and improves users' quality of experience."**

The advantages of employing delaying strategies are evidenced in some previous research studies (LI; PU; AHAMAD, 2004; HUSNA; PHITHAKKITNUKOON; DANTU, 2008). Proposed anti-spam frameworks slow down spammers by adding delay to email delivery. Furthermore, when the cost of retrieving contents from a system is considerably high, it is worth adding artificial delay in order to avoid wasting resources. For example, in file sharing systems, the cost in terms of bandwidth and time spent to download a file may be too high; hence, users wait to increase the chances of retrieving expected contents. This trade-off between waiting times and users' quality of experience is investigated in detail throughout this thesis.

The solution devised in this thesis may be instantiated in any kind of CDS. However, due to the use of delaying strategies, its applicability should be carefully considered in time-constrained systems, e.g., live streaming. We designed our solution for general CDS, being file sharing and streaming systems two representative examples. The set of experiments carried out in this thesis comprises both real tests and simulations. During the progress of this work, the overall solution was improved (through iterations) to become more robust and consider different users behavior, always focusing on reproducing real environments. Thus, we decided not to employ the same values for input parameters and rather approximate the evaluation scenarios to mimic important facets present in the wild. For example, to evaluate FUNNEL and prove the feasibility of the conservative strategy, we reproduced a BitTorrent community in a controlled testbed. Similarly, to evaluate DÉGRADÉ, a sequence of tag assignments needed to be incorporated to the solution so as to represent annotations of users. Furthermore, user's arrival rates were updated to be in accordance to more recent measurements found in the literature.

1.1 Contributions

The incremental steps just introduced unfold in three main contributions which we detail next.

- Design of a conservative strategy based on binary votes to fight pollution in CDS.
  The solution was instantiated and evaluated on a real BitTorrent infrastructure (SANTOS et al., 2011).

- Design of a robust strategy that employs user generated content (textual annotations) to circumvent pollution.
  The solution was instantiated and evaluated using an extensive set of annotations provided by the popular Delicious online bookmarking system (SANTOS et al., 2013).

- Development of an analytic model that employs delaying functions to mitigate general massive attacks.
  The solution was instantiated and evaluated on different use cases (SANTOS et al., 2013).

1.2 Organization

Chapter 2 first presents an overview on content CDS and depicts the most popular systems in two different fronts: file sharing and streaming systems. Then, a list of derived commonalities is presented and described. Finally, the last part of this chapter provides technical background on the content pollution and massive attacks, and discusses related work.

Chapter 3 introduces the first solution proposed in this thesis, named FUNNEL. The conservative model is described in detail and an experimental evaluation is conducted using a real BitTorrent infrastructure.

Chapter 4 presents DÉGRADÉ, an extension to the previous solution to lessen the subjectiveness problem related to content pollution. This novel solution is evaluated using crawled data from an online bookmarking system and the results are discussed under different perspectives.

Chapter 5 formally defines pollution, based on previous results, as a massive attack. Subsequently, an analytic model to tackle the problem is presented and validated using different use cases.

Chapter 6 summarizes the contributions and key findings of this thesis, and presents prospective directions for future research.

2 BACKGROUND AND STATE OF THE ART

Most popular content distribution systems emerged almost a decade ago. Since the standardization of MP3 files, in 1991, several file sharing systems appeared. Audiogalaxy was a precursor of those systems, released in the late 90s. It was designed exclusively to share audio files and quickly attracted the attention of recording industry associations. Napster was another system also released with similar purposes. Both systems were discontinued due to copyright infringements and their services were definitively shut down some years later. The collapse of previous systems motivated some others to appear. Around 2000, Kazaa and Gnutella emerged as precursors of a new generation of file sharing systems. They started using more decentralized approaches to avoid being caught by recording industries, who concentrated efforts for years to shut them down. Although both systems are still running, the number of users has dropped dramatically last years since lawsuits started being settled against piracy in the world (PC Magazine, 2011). In 2001, BitTorrent was introduced as a novel protocol to efficiently distribute large amounts of data and quickly gained attention from anti-piracy companies. A study carried out in 2009 revealed that BitTorrent was the most popular protocol used to share content in the world (SCHULZE; MOCHALSKI, 2009).

As part of an evolutive process, each of these systems was created to circumvent practical limitations of predecessors. In the beginning, applications were created to enable users to share audio files in MP3 format. Along with the evolution of computers and storage devices, users also started sharing different kinds of content, beyond audio. The need for more flexible services fostered the design of more robust systems. Users wanted to be able to search and download any kind of content efficiently. Continuous efforts from developers and researchers have been made towards more sophisticated file sharing systems (ANDROUTSELLIS-THEOTOKIS, 2002).

Advances on communication technologies motivated other interactive applications. As an example, streaming systems stand out as a novel reference to disseminate media on the Internet. Audio and video are the most common categories which benefit from streaming systems. Streaming may disseminate previously recorded or live content. Several systems emerged in the recent past, such as Spotify (KREITZ; NIEMELA, 2010), CloudStream (HUANG et al., 2011), SplitStream (CASTRO et al., 2003), TVAnts (SILVERSTON; FOURMAUX, 2006), SopCast (LU et al., 2009), PPStream (JIA; LI; CHEN, 2007), PPLive (SPOTO et al., 2009), Tribler (POUWELSE et al., 2008), and LiveShift (HECHT et al., 2011). They implement different features and policies according to their targeted audience. Some of them support only a limited number of streams (also known as “channels”), which are broadcasted by certified companies, while others allow also regular users to broadcast content.

As systems became more flexible, e.g., by allowing users to create their own channels,

some security issues have been exposed. In the context of content generation, media with poor quality and imprecise description became possible. Moreover, selfish behaviors were evidenced in many CDS (LIANG et al., 2005). These problems highlight the importance of designing robust countermeasure mechanisms while considering advances in CDS. Although we present fruitful discussions about designs and systems, the reader familiar with content distribution systems and massive attacks may feel comfortable to skip this chapter.

Organization. The remainder of this chapter is organized as follows. Section 2.1 describes several content distribution systems, considering both file sharing and video streaming, with respect to a predefined set of dimensions. Section 2.2 discusses content pollution problems and analyzes previous work related to this thesis. Finally, Section 2.3 closes the chapter with a summary.

2.1 Content distribution systems

This section presents a detailed discussion about content distribution systems. First, a taxonomy is provided to classify CDS with respect to different dimensions (Subsection 2.1.1). Second, the category of file sharing systems is discussed according to the predefined dimensions (Subsection 2.1.2). Then, streaming systems are presented and discussed (Subsection 2.1.3). Finally, section closes with a discussion about key aspects of studied systems (Subsection 2.1.4).

2.1.1 Classification and dimensions

The study carried out in this section investigates CDS under different dimensions. We first describe them and then classify some of the most popular systems according to their nuances.

- *Architecture.* The architecture of content distribution systems may be classified according to three different structures. The first one comprises the popular client-server architectures. In these architectures, the service is provided exclusively by the server(s), whilst clients are only regular consumers. The second one is called Peer-to-Peer (P2P) architecture. In these architectures, consumers are also capable of supporting/improving the service by sharing their available resources, e.g., storage, computing power, and network bandwidth. The third, and last, architecture considered in this study comprises a hybrid infrastructure. In this case, the list of features provided by the service is instantiated on the server and partially also on the clients.

- *Usability.* The term usability refers to the ease of use and the learning process related to a system. Although it comprises a wide research field that relates distinct areas of investigation, we limit this study to three main aspects of usability. The first one is the efficiency to use, which measures the time to accomplish regular tasks. The second one is the learning process, which represents the time to get used to the system. The third one is the user satisfaction, which measures his feeling about the system.

- *Operation.* The operation of a content distribution system is analyzed in two different steps. First, it is considered the system configuration in terms of parameters and infrastructure requirements. Second, it is considered the complete deployment

procedure to launch the system. Both steps allow estimation on how much effort is required (from users and/or administrators) to keep systems up and running.

- *Efficiency.* The efficiency of content distribution systems can be measured using several metrics. Efficiency in terms of time may represent the latency until the last user receives the entire content. It may also represent how efficient the connection management is, avoiding redundant traffic and bandwidth waste. Furthermore, efficiency may represent the bandwidth allocation of users. The more bandwidth used, the more efficient the system would be.

- *Effectiveness.* This dimension measures if a CDS provides the expected content to users. The system is supposed to provide good experience to users when they search for a content, download and evaluate it. In the context of this study, CDS are analyzed and compared assuming that user's queries are always precise. This mitigates potential subjectiveness concerning users behavior and allows a fair comparison among systems.

- *Security.* It is recurrent that content distribution systems expose information about users. In some cases, every user can access a list containing IP addresses of others. In other cases, it is possible to access user's personal information. Depending on systems design, user's privacy can be seriously compromised. Besides privacy, another relevant security issue to be considered in content distribution systems is related to content integrity. Thus, the data received by users should be the same originally published.

- *Legal aspects.* This dimension refers to the nature of content disseminated in the systems. Since most of CDS is used to disseminate video content (SCHULZE; MOCHALSKI, 2009), recording industries are directly affected by unauthorized distribution of copyrighted material. The goal is to analyze content distribution systems from the legality perspective.

- *Economic aspects.* Two economic aspects of content distribution systems are analyzed in this dimension. First, positive aspects discuss how companies make money out of these systems. Second, negative aspects are presented when those systems are used to illegal purposes.

2.1.2 File sharing

The first kind of CDS discussed is file sharing systems. These systems enable users to share contents over the Internet. To be considered a file sharing system, it should support both store and retrieve operations. In the following, we describe the most popular systems with respect to the aforementioned dimensions.

MegaUpload and Rapidshare

There are many systems like MegaUpload and Rapidshare, e.g., boxify (BOXIFY, 2012), minus (MINUS, 2012), and justbeamit (JUSTBEAMIT, 2012). These systems are known as file hosting services and are part of the first class of systems used to distribute content on the Internet.

Architecture. File hosting services use client-server architecture. Since these services are provided by companies, it is hard to know how they organize their internal infrastructure. To cope with large amount of data and traffic, they use clusters of machines and hard drives. It is important to notice that it is possible to have a client-server architecture using replicated servers (mirrors).

Usability. In order to use these services, users just need to access a website or a proprietary (and specific) application and select the file to upload. Most of the services do not require registration, so users upload a file and retrieve a link for further downloads. Typically, services do not limit the upload rate and the whole process is easy and straightforward. However, to retrieve the contents, users need to wait a while and experience low download rates intentionally imposed by some providers to motivate users to upgrade their accounts. Moreover, some providers define the maximum amount of data that users can store on the system.

Operation. The infrastructure is run by service providers. They offer storage and network resources to distribute contents. Although HTTP is normally used, sometimes users need to download and install a specific software to interact with the service.

Efficiency. Since these systems use client-server architectures, the server needs to send the entire file to every new request. This generates a lot of redundant traffic and significantly increases overall costs. To reduce the latency between the server and users, and also the inter-domain traffic between Internet Service Providers (ISPs), mirroring is a technique frequently used by these services.

Effectiveness. Users publish files and describe them. In this kind of services, only the publishers can provide metadata. This may lead to imprecise and heterogeneous descriptions. Therefore, using storage services to find contents may be a cumbersome task, also because some of them do not even provide search mechanisms. To circumvent that, blog communities index contents by providing links and additional descriptions to files.

Security. Depending on the system, a registration may be required in order to upload files. If so, the system stores information about all uploaded files, as well as user's personal data – e.g., email address and name. Moreover, users' IP addresses can be associated with upload activities and, eventually, be exposed by services. On the other hand, if the system does not require a registration, the only exposed information is the IP address related to a certain activity.

Legal aspects. Service providers keep track of all uploaded files and can easily react in case of copyright infringements. However, they use passive policies and remove illegal contents upon request.

Economic aspects. In general, file hosting services make profits from downloaders and uploaders activities. In the former, services limit downloading rates of users and/or impose delays to access files. In the latter, services define a maximum size per file and a quota per user. In order to get more privileges, users need to pay a subscription, which may vary according to the benefits offered.

Peer2Mail

These systems use mail servers to store content. Users need to run a local application that splits the files (in order to fit servers rules) and use private accounts to store these data. In order to retrieve the content, the same application must be used to download the parts of a file and merge them.

Architecture. Since it uses a user account to store data in a specific mail server, the

architecture can be classified as traditional client-server. The service does not require any modification on server-side and can be used in many mail servers. The architecture is already in place and requires only a client application to handle splits and merges of files.

Usability. In order to use these services, users must download a third-party software and configure it to use a specific email account to store data. Once logged in, users can list the files uploaded to his account and easily download them. To share contents with other users, the email account must be also shared.

Operation. There is no need for a service configuration or deployment. One clear advantage of these systems is that they require a client-side software and everything works transparently. The most effort for the users is to configure an already existing email account to store files.

Efficiency. Since mail servers are used as storage systems, all traffic goes through a centralized point. Although this produces high costs for servers, users can take advantage of powerful infrastructures and download content at high speeds.

Effectiveness. In these systems, contents are described only by their filenames, which demands additional indexing mechanisms. In order to enable users to find contents, portals and forums need to publish content descriptions and the email account where files can be downloaded. There is no guarantee that content descriptions and files match, so users must trust on indexing systems.

Security. These systems rely on the infrastructure of mail servers, hence, always require a user account to work and store data. Thus, every uploaded content is associated with a specific account. Another security issue concerns file integrity. Since large files need to be split to fit mail servers, potential problems may arise during the merge process, when users download the files. Particularly, it is difficult to detect which part of the file is corrupted due to the absence of hashing schemes.

Legal aspects. The architecture of these systems demands a laborious work to detect illegal content. The published contents are stored in personal email accounts, in which a user may claim that they are for personal uses only. The terms of service vary from one mail provider to the others and the process to delete user's email messages is not trivial. Moreover, users can simply change their account passwords and the infrastructure is settled again. It is difficult to anti-piracy companies monitor user's email account in order to find copyrighted content.

Economic aspects. The model employed by these systems does not provide a clear way to make profit. Conversely, the misuse of mail servers increases storage and infrastructure costs. Client applications display advertisement to users, but since there is a lack of description for contents, it is difficult to show effective/personalized advertisements.

Napster and Audiogalaxy

This class of content distribution systems was the first not to store files in a centralized point. Instead, they stored files across user's machines and maintained a centralized index to assist users in finding contents. Due to this centralized index, many anti-piracy agencies went to court and forced them to shut down. Nowadays, these systems adjusted their business model to offer legalized content and/or services. The scope of this section is limited to the old versions of these systems.

Architecture. These are considered the first peer-to-peer systems to gain popularity on the Internet. Instead of traditional client-server architectures, in which each client must download the content from the server, clients (called peers) connect directly to each other

in order to download the files. This novel approach reduces the overhead on server-side and replicates files across the network.

Usability. These systems were originally designed to share MP3 files. In order to use the system, users need to download a client application, search for any music by typing a query, and choose one to download from the result set.

Operation. The whole system relies in a unique central point to work. This central point is responsible for indexing files and enabling users to find each other. The client applications are configured with a default server address and users just need to connect and search for music. Moreover, users need to define a shared folder to publish and store downloaded contents. When the application is launched, it announces the files in the shared folder to be indexed by the server.

Efficiency. Users play an important role on these systems by storing and uploading contents. The central server only needs to index contents instead of storing them. This technique saves upload bandwidth on the server and starts using users' bandwidth. Therefore, the efficiency of the system depends on users' capacity. Since the content is downloaded from a single user at a time, the rates are limited according to sender/receiver capacities. Besides, the system performance becomes dependent of users' reachability, i.e., users behind firewall or NAT most likely are not able to receive income connections, which limits the overall performance.

Effectiveness. Since these systems were originally designed to share MP3 files, users can search using file names and metadata. Users are responsible for providing metadata according to their own perceptions. This autonomy generates heterogeneous descriptions and potential fake files. Moreover, data exchanged may be corrupt due to communication fails or malice. Should this happens, users need to download the entire file again.

Security. The addresses of users are stored on the server in order to enable direct connections between them. Furthermore, since users must connect to each other, their IP addresses are exposed in the network, which might be undesirable by some users. The restrictions about the type of files supported by these systems keep them reasonably safe against virus and malwares.

Legal aspects. These systems work as a central directory of files. As they got popular in the web, more users started abusing them and sharing copyrighted contents. This fact attracted the attention of recording industry, which started fighting in court to shut the services down. After losing the case, their central servers were forced to stop.

Economic aspects. The sharing activities in these systems produced a huge loss for recording industries. Besides this negative impact on the market share of companies, these file sharing systems base their revenue on selling advertisements both on websites and client softwares.

Kazaa, eDonkey, Gnutella, and BitTorrent

This class of applications emerged as an alternative to decentralize systems load and improve performance. Different strategies were introduced in order to speed up content dissemination. Instead of downloading a content from a single user, these systems implemented parallel transfers by dividing files into smaller parts and requesting them from multiple users in the overlay. By adopting decentralized infrastructures, these systems attempt to hamper anti-piracy actions to mitigate content sharing. Except for Kazaa, which changed to a streaming service (like Napster), all the others are still active as originally proposed.

Architecture. These systems employ mesh architectures where peers contact each other to exchange piece of files (chunks). Every peer has incoming and outgoing connections to a set of other peers. They typically form random or small-world topologies to enable fast and robust communication (AL-HAMRA; LEGOUT; BARAKAT, 2007; DALE et al., 2008; FAUZIE et al., 2011; KRYCZKA et al., 2011). Since peers need to share chunks, the more neighbors and potential interesting peers they have, the faster the downloads can be (CAPOTA et al., 2011).

Usability. In order to use these systems, users need to find the desired content. Most of the systems (excluding BitTorrent) offer native search mechanisms in the protocol level, i.e., users can open the software, type their query, and choose any content to download. However, BitTorrent protocol does not offer this feature. Instead, users need to search for a metadata file (torrent) through web portals, for example, and load it on their preferred software. These torrent files contain references to the real content and enable peers to connect to each other. Portals play an important role in aggregating torrents and users feedback about them.

Operation. Once users decided which file to download, the next step is to find peers which have chunks of that content. Kazaa and Gnutella use a flooding strategy to reach a huge number of potential uploaders. The ones who have the content announce themselves as available. To avoid overloading the network with messages, protocols set a time-to-live (TTL) around 7 hops. Conversely, eDonkey and BitTorrent offer infrastructures to find neighbors given a content hash, which may be found by other means as explained before.

Efficiency. Since these systems rely on the neighbors capacities and the overlay topology, incentive mechanisms to enforce users to contribute are of paramount importance. The natural behavior of users is to download as fast as possible, saving their resources as much as they can (HUGHES; COULSON; WALKERDINE, 2005). The first version of Gnutella protocol did not have any incentive mechanism, which resulted in about 70% of free-riders in the system (ADAR; HUBERMAN, 2000). Proposed solutions tried to employ the amount of upload and download per user, within time slots or not, to foster contributions. This strategy is called tit-for-tat (TFT) and advocates that the more users contribute, the more they should receive.

Effectiveness. In this kind of decentralized systems, users behave arbitrarily and publish contents without any moderation. Since there is no central point where files are indexed, it becomes easy to flood the system with fake data. Moderation efforts are difficult to maintain due to the lack of central points. One of the advantages of using chunks to disseminate contents is that, in case of corrupt data, only the invalid chunks need to be downloaded again.

Security. User IP addresses are spread through the network. In order to start downloading contents, peers need to find the location of other peers. By definition, a mesh topology comprises a set of connections between pairs of users. Although some advances were presented in the literature (BLOND et al., 2011), privacy is still an issue in most P2P systems (BLOND et al., 2011). Content corruption is also a well-known problem in file sharing systems (KONRATH; BARCELLOS; MANSILHA, 2007). However, hashes for the entire content or their chunks (COHEN, 2003) and reputation mechanisms have been used to protect the integrity of files (BARCELLOS et al., 2008).

Legal aspects. These systems, like most file sharing systems, have been targeted by anti-piracy companies due to the huge traffic of illegal copyrighted contents. Kazaa services were shut down by court and emerged again as a legal streaming service. The other services are still in place and are frequently used to disseminate copyrighted content.

Economic aspects. Kazaa system concentrates its revenue on selling subscriptions. Users pay regularly to access a library of online songs. On the other hand, eDonkey, Gnutella and BitTorrent applications show advertisements to their users. Except for Kazaa, these systems are frequently used to share copyrighted content without authorization, negatively impacting recording companies (SCHMIDT et al., 2012).

2.1.3 Streaming systems

Streaming systems are defined as systems capable of distributing media, e.g., audio and video, to a set of users through a network. According to Liu, Guo, and Liang (LIU; GUO; LIANG, 2008), they can be classified into two categories: on-demand and live. In an on-demand system, users enjoy the flexibility of consuming whatever content whenever they want. The playback of the same content on different users are not synchronized. In contrast, during a live streaming session, the same piece of content is disseminated to all users in realtime; hence, the content playback on all users is synchronized. Examples of these systems are discussed in the following.

Streaming on demand

- *YouTube and Hulu*

  Architecture. These systems use a client-server approach to disseminate content on the Internet. They were originally created to run on users' browsers and, later, their services were also released for mobile devices. The back-end of these systems uses clustered machines to process users' requests and store contents.

  Usability. In order to benefit from these systems, users need to open their browser (or access YouTube/Hulu specific applications) and go to the desired website. Both services provide a searching mechanism where users type queries and look for contents. There is also recommendation systems to suggest featured and sponsored videos. In some cases, users need to install plugins or extensions on browsers to be able to watch online content.

  Operation. The requested media is loaded on server and sent to users. The traffic is encapsulated in HTTP packets and relies on specific software on client side to be played. In the case of YouTube, the majority of contents is generated and uploaded by users, whilst in Hulu the contents are essentially provided by big players.

  Efficiency. The efficiency of these centralized services is limited by the providers infrastructure. Since entire contents are stored on their administrative domains, storage devices, processing units, and network bandwidth may support only a limited load. Unless they migrate to a more decentralized architecture, extra equipments need to be purchased in order to extend their capacity.

  Effectiveness. YouTube is a service that basically depends on contents generated by users. Therefore, its catalog of videos is susceptible to malicious interferences. Uploaded videos are described by their publishers, which potentially generates imprecise and controversial descriptions. Publishers are interested in attracting audience and, therefore, have incentives to employ popular/interesting words. Often, these words do not match the content itself and cause a bad quality of experience for users (BENEVENUTO et al., 2008, 2009). Conversely, contents published on Hulu are described by their producers or people with enough knowledge about those pieces of content. Since it is a moderated platform, quality of experience tends to be

higher than in YouTube, along with the cost of moderating every published content.

Security. These systems are maintained by private companies, which provide their own security policies. Contents are stored on robust data centers and under companies' responsibility.

Legal aspects. Every content uploaded to YouTube needs to be aligned to their terms of service. Although they use algorithms to detect and deny copyrighted content, many illegal material are daily submitted to their servers. In those cases, media producers and artists should report to a team of moderators, who acts to avoid law infringements. Similar problems do not happen to Hulu, because contents are previously licensed close to regulatory agencies before being published.

Economic aspects. The business model used in these systems is based on advertisements. Additionally, they offer users a paid subscription that grants more space and longer videos on YouTube, for instance, and packages with more TV shows and movies on Hulu. Part of the revenue is naturally converted to copyright companies.

- *Tribler and Vuze*

  These systems are built on top of BitTorrent protocol and, hence, present most of its characteristics. A key feature implemented in these systems is the ability to increase download priority of first chunks. This is of paramount importance to enable players to reproduce the media before the download is complete. This modification degrades the efficiency of BitTorrent protocol, since the Local Rarest First (LRF) policy is severely affected (LEGOUT; URVOY-KELLER; MICHIARDI, 2006; SHAH; PARIS, 2007).

Live streaming

- *SplitStream*

  Architecture. This system uses a tree-based architecture to push content to users. The stream is split into stripes and multicast each stripe using a separate tree. SplitStream relies on a structured P2P overlay to construct and maintain these trees.

  Usability. Since this is a more academic work, the SplitStream prototype is not available for downloading on the Internet.

  Operation. The content is generated by the broadcaster and split into stripes. Users join as many stripes as they wish to receive and specify an upper bound on the number of stripes they want to forward.

  Efficiency. Since SplitStream uses a tree-based approach, the task of maintaining the overlay is tough. The main problem is that updating the topology of the trees becomes harder in the presence of churn. Furthermore, users who stay in the leaves of a tree do not contribute to the others. Thus, they should preferably be in different positions when connected in other trees. Another problem related to the tree-based topology concerns bottlenecks. Users with low upload bandwidth might negatively impact all their descendants in the tree overlay.

  Effectiveness. Although there is no official SplitStream release, its proposed architecture allows regular users to create and broadcast their own content. The prototype does not provide any means to search for contents. However, the effectiveness of the system is severely affected in the absence of mechanisms to avoid content pollution.

Security. Isolated selfish behaviors are tolerated by this system due to the use of multiple trees, but a coordinated eclipse attack may be enough to degrade system's overall performance.

Legal aspects. Since this system allows users to broadcast their own content, copyright infringements remain an issue.

Economic aspects. The prototype is part of an academic work and has no direct economic interest.

- *PPLive, PPStream, and SopCast*

  Architecture. These systems' architecture is based on a mesh-pull overlay, i.e., chunks of data are sent under request. Each overlay represents a channel in the system and users may participate in many of them at once.

  Usability. In order to watch a channel, users must use a proprietary client. One or more channels can be selected and the transmission is started.

  Operation. A central server is responsible to maintain a list of available channels, which is downloaded when clients are started. SopCast allows users to broadcast their own channels. When users select a channel, the client queries a membership server for initial peers participating on that channel. In order to gather more peers, further queries can be performed to this server. Moreover, a gossip protocol is used to exchange list of peers.

  Efficiency. These applications maintain a set of servers dedicated to upload content. However, to reduce their load, they require users to share their upload bandwidth. By default, clients contribute their full available capacities and do not prioritize any other client. This lack of incentives encourage users to "free-ride". To circumvent that, Piatek et al. (PIATEK et al., 2010; MOL; EPEMA; SIPS, 2007) have proposed the use of contracts (i.e., receipts) to certify data transfers among peers in PPLive. These contracts are employed to prioritize the ones who contributed most when selecting neighbors.

  Effectiveness. Since channels are maintained by PPLive and PPStream administrators, there is a considerable effort to provide precise descriptions to users. This approach is considered effective for low number of channels, but may not scale when this number increases. In contrast, channels are also created and described by users in SopCast, which may lead to many polluted content.

  Security. Mesh-based approaches are exposed to some privacy issues. Since discovering peers is important to keep systems working efficiently, users' IP addresses are exchanged and flooded in this type of network. Besides, attackers can connect to many IP addresses and start uploading junk data. This malicious behavior may overload victims and tamper with their video reproduction.

  Legal aspects. In PPLive and PPStream, channels are moderated by administrators and regular users cannot simply create new ones. Thus, copyright infringements are not a problem in those systems. However, SopCast users may abuse the system to disseminate copyrighted content on their own channels.

  Economic aspects. The business model used in these systems is based on advertisements and special subscriptions offered to their users. In addition, SopCast sells hardware to host their channels and services.

2.1.4 Discussions

In the previous subsections, we have shown that centralized infrastructures may offer robust and reliable services to users. However, the costs to maintain redundant machines and network links are high. Furthermore, a laborious work is required to keep the system free of fake content uploaded by users. To reduce costs, more decentralized architectures have been proposed by researchers and developers. Therefore, users started playing an important role in the system by sharing their resources with others.

When the first CDS appeared, systems required users to be altruistic, hence, incentive mechanisms were not in place to enforce them to share their resources. Nevertheless, most of users behave selfishly and overall systems performance were severely affected (LIANG et al., 2005). This behavior fostered the development of mechanisms based on reciprocity to incentivize users to cooperate. Reciprocity-based mechanisms are commonly employed when users of a system have mutual interests on each other. Reputation mechanisms and BitTorrent's TFT are examples of popular techniques based on reciprocity to classify users and/or contents in a system.

With the advent of live streaming systems, most of the incentive mechanisms became unsuitable and/or ineffective. Due to live aspects of the transmission, the widely-used bilateral incentive strategies could not be effectively applied to the live streaming environment (PIATEK et al., 2010). Different approaches were proposed to artificially introduce mutual interests, for example, by splitting the streams (CASTRO et al., 2003; MOL; EPEMA; SIPS, 2007) or periodically changing the overlay topology (MAGHAREI; REJAIE, 2007). In spite of clear improvements on live streaming systems, they still face overload problems to provide short delays and low latency (HEI et al., 2007; SILVERSTON et al., 2009).

To overcome the limitations of purely decentralized systems, hybrid architectures have been conceived. This approach tries to use as much resource from users as possible, but also provide dedicated infrastructures to guarantee certain level of QoE to users. Spotify (KREITZ; NIEMELA, 2010) is a popular audio streaming system that uses hybrid architectures to broadcast content. This successful case was also applied in the context of file sharing systems (ANDRADE et al., 2007; ROCHA et al., 2009). However, as highlighted in previous sections, content pollution problems remain an open issue in most CDS. This is the main object of study of this thesis and is discussed in detail next.

2.2 State of the art

This section first elaborates on the content pollution problem to better motivate the investigation carried out in this thesis. Due to the importance of massive behaviors in this context, massive attacks are also introduced. The most prominent solutions proposed by the scientific community to tackle these problems in CDS are also discussed in detail in this section. Several attacks and countermeasures are investigated in the context of file sharing and streaming systems.

2.2.1 The content pollution and massive attacks

Given the overview on content distribution systems, this subsection describes in detail the specific problem related to pollution. Moreover, the problem is generalized to consider the wide class of massive attacks. There are two main types of pollution in CDS: content corruption and decoy insertion. In the former, malicious users replace parts of a

content with junk data in order to violate its integrity. In the latter, malicious users continuously upload contents with incorrect descriptions (metadata). Isolated polluters are easily tolerated by many systems, however, content pollution becomes a serious problem when combined with massive attacks.

Corruption and decoy insertion attacks may be present in file sharing (DHUNGEL et al., 2008) and also in streaming systems (YANG et al., 2008). Dhungel et al. verified the existence of corruption attacks in BitTorrent communities (DHUNGEL et al., 2008). In streaming systems, malicious users may try to upload junk data or modify authentic content to promote advertisements, for example. Evidences of decoy insertions were reported by administrators in BitTorrent communities (TORRENTFREAK, 2009). Content with incorrect metadata were also reported in streaming systems to attract user's attention or tamper with their experience (BENEVENUTO et al., 2012).

To understand the seriousness of the problem, we carried out a measurement study that estimates the amount of polluted content (decoys) available for download in BitTorrent communities. It is worth stating that we do not take sides on the P2P file sharing debate; instead, we present evidence of pollution dissemination in BitTorrent file sharing communities.

The measurement study consisted in querying large BitTorrent communities for popular movies. Since it is not feasible to query for every published content, we focused on movies whose release date is 2010. A list with the top-250 movies according to IMDb (IMDB, 2012) was collected and their titles were searched at the isoHunt BitTorrent community. The results for each query were matched against a database available at Bitsnoop BitTorrent community (BITSNOOP, 2012), as explained below. Queries with no results were removed from the analysis.

Bitsnoop employs an automatic mechanism to determine the status of content "referred to" by torrent files. The mechanism, named FakeSkan, basically retrieves and analyzes torrents that present suspicious information (e.g., names, file sizes and structure). In addition to analyzing torrent data, the mechanism also considers users' votes to set the status of a torrent. When inconsistencies arise, FakeSkan presents warnings to users about to download the content. Internally, it operates as follows. A torrent may be initially flagged as VERIFIED or FAKE depending on whether it comes from a secure source (e.g., previously verified by other community administrator) or there are strong signs that the torrent may be polluted (e.g., torrent file name referring to some book collection but containing executable files). Otherwise, the torrent is automatically classified as GOOD or BAD according to user feedback, and eventually changed to either VERIFIED or FAKE by an administrator. In the absence of feedback from users, the torrent status is set to NONE. Thus, the status that results from a query about a torrent may be one of the following: VERIFIED, GOOD, NONE, BAD, FAKE, or NOTFOUND. The latter case happens when FakeSkan is unaware of the torrent.

In our analysis, torrents returned by queries to the 250 most popular movies in 2010 were inspected by Bitsnoop. Table 2.1 summarizes the query responses grouped by torrent status. The results show that 20% of torrents are flagged as FAKE or BAD, i.e., almost 2.4 times the number of VERIFIED and GOOD copies, which account for roughly 27% of the downloads. Our findings are consistent with a recent study, which concludes that 30% of torrents are published by anti-piracy agencies or malicious users, and this proportion of torrents are responsible for 25% of the downloads (CUEVAS et al., 2010).

The summary of torrent status presented in Table 2.1 gives an overview of the pollution scenario. However, we would like to know the impact of pollution on users' perspec-

Table 2.1: Torrent clusters with 90% confidence interval for averages

Torrent status	# of copies	Avg. torrent len. (MB)	Avg. # of seeders	Avg. # of leechers	# of downloads
NOTFOUND	22,794	775.226 ± 5.734	1.100 ± 0.378	1.247 ± 0.147	314,673
VERIFIED	6,901	1557.930 ± 35.037	94.110 ± 10.147	38.544 ± 5.015	11,338,693
GOOD	83	860.338 ± 66.751	24.096 ± 20.294	8.494 ± 6.041	137,404
NONE	36,694	823.835 ± 5.524	7.329 ± 0.828	3.421 ± 0.257	2,643,607
BAD	4,055	748.813 ± 7.874	19.254 ± 5.010	4.584 ± 1.105	215,189
FAKE	12,707	771.105 ± 2.697	97.488 ± 15.106	37.541 ± 14.024	5,225,590

tive, represented by the probability that a user finds a polluted copy when searching for contents. In order to understand this impact, we plotted a histogram of torrent status for queries, averaged in groups of 20 movie titles, as shown in Figure 2.1. Nearly 25% of the queries resulted in lists with more than 20% of the torrents flagged as FAKE, hampering the process of finding the desired content. In order to obtain a more precise estimate of the amount of traffic generated by polluted contents, we investigated at isoHunt the size and number of downloads performed with each copy flagged as FAKE.

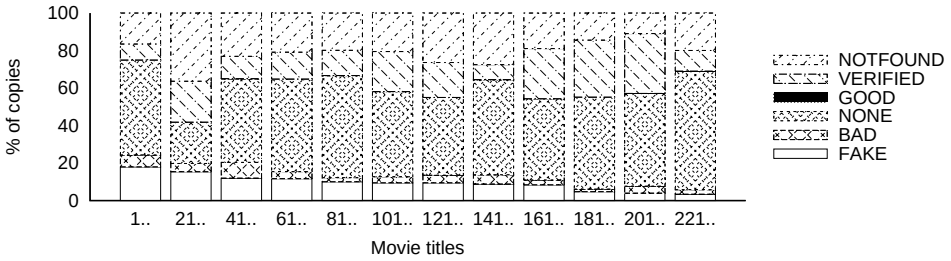


Figure 2.1: Histogram of torrent status per groups of 20 movie titles, sorted by the proportion of copies marked as FAKE.

Figure 2.2 shows the complementary cumulative distribution function (CCDF) of the number of completed downloads of copies considered FAKE. A pair *(x,y)* in this plot means that the probability that a copy is downloaded at least x times is equal to y. Since the 94% (11,956) less frequently downloaded torrents were obtained at most 10 times (reflecting a curve close to y-axis), the graph was adjusted to focus on the most frequently downloaded torrents. The plot shows that ∼3.8% (500) fake copies were downloaded over 5,000 times. This number is even higher if we consider the ∼2% (250) most downloaded copies, which were obtained more than 10,000 times. A small portion of fake torrents were downloaded more than 15,000 times. In summary, in the context of the 250 most popular movies of 2010, the copies flagged as FAKE summed up to 5,225,590 downloads.

We estimated the (negative) impact that these useless downloads had on the amount of data traffic generated on the Internet. Downloads of copies flagged as FAKE accounted for ∼3.5 PB (3,572 TB) of data traffic; this value increases by approximately 149 TB if we also consider the downloads of BAD copies. These results indicate that pollution leads to a significant and unnecessarily waste of client and network resources (such as time and network bandwidth). More importantly, they highlight the importance of devising mechanisms to control pollution.

2.2.2 Related work

This thesis is related to three main research fields: content pollution, massive attacks, and conservative strategies. Incremental steps of our research traversed these three general areas, which are detailed next.

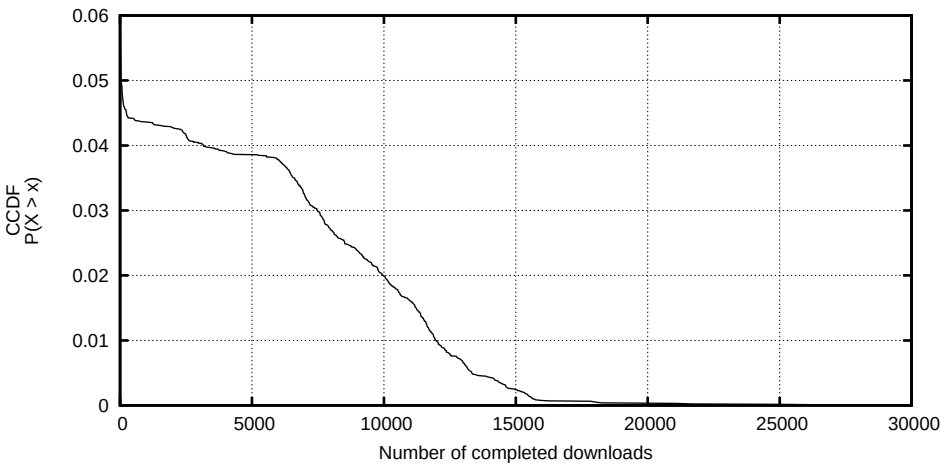


Figure 2.2: CCDF of the number of completed FAKE copy downloads

Content pollution approaches

In this subsection we cover the most prominent solutions proposed so far by the scientific community to fight pollution in CDS. In order to organize the discussion, we first focus on the decoy insertion problem and then analyze the problem of mislabeled contents.

Malicious decoy. Solutions to fight decoy insertions in CDS are fundamentally based on reputation mechanisms. The set of solutions proposed up to date may be classified in three categories. The first one tries to assign a reputation score to shared contents. Similarly, the second one builds users' reputations to isolate malicious publishers. The third one combines the other two and uses both contents and users' reputation to mitigate pollution.

In the first category, Walsh and Sirer (WALSH; SIRER, 2006) present a reputation-based mechanism, called Credence, which assigns scores to contents in order to govern their dissemination in the system. When users download a content, they evaluate it and assign a numeric vote from -1 to 1. However, before users start downloading a content, they gather testimonials from neighbors to take a decision. Each vote is weighted by a correlation index calculated by analyzing previous votes to common downloads between users. Credence relies on the assumption that honest users issue similar votes when establishing the authenticity of a given content. In this case, users having higher voting similarity are more trustworthy than others having lower similarity. The evaluation of the model, using the Gnutella network, led to satisfactory results. This strategy, however, does not prevent newly published, polluted contents to be widely disseminated when there is little knowledge available to form an opinion about their reputation. This is worsened when new polluted contents are quickly published.

Built on top of the idea of traditional reputation-based mechanisms, Cai et al. (CAI et al., 2009) proposed a holistic mechanism which combines users' feedback with content-associated information and statistical data reflecting the diffusion state of contents to defend against pollution attacks. The mechanism provides a set of strategies integrated to contents lifecycle (publication, search, selection, and download) that help users on distinguishing polluted versions. Since authors assume that (*i*) malicious users cannot spoof their identities and (*ii*) the routing protocol is reliable, problems related to vote integrity and inappropriate content descriptions remain unsolved.

In the second category of reputation mechanisms, Eigentrust (KAMVAR; SCHLOSSER; GARCIA-MOLINA, 2003) is employed to establish a global trust (i.e., reputation) value for each peer based on their upload records, which could then be used to select reliable sources. One practical limitation of Eigentrust is the need of a set of pre-trusted users to operate accordingly. To circumvent this issue, Scrubber (COSTA et al., 2007) proposes a distributed voting protocol, in which peers combine individual experiences and testimonials from other peers to calculate reputation scores. The individual experience increases whether a peer has good interactions with some other, and decreases otherwise. Moreover, testimonials of neighbor peers are requested periodically to create a more robust mechanism. Thus, a peer is considered trustworthy if its final reputation is above a pre-defined threshold. In order to avoid vote manipulation from neighbors, Vieira et al. propose SimplyRep (VIEIRA et al., 2012), a reputation mechanism based only on local observations. Furthermore, SimplyRep introduces a dynamic threshold to classify peers as malicious considering the system state: tempest or calm. An issue common to the solutions presented so far lies in their vulnerability to *whitewashing* attacks (JOSANG; ISMAIL; BOYD, 2007), since newcomers and newly published contents start with a neutral reputation.

To circumvent that, researchers proposed solutions that combine both content and peer reputations to protect CDS against pollution. XRep (DAMIANI et al., 2002) is one of those mechanisms that employs distributed voting algorithms to identify potential authentic content. XRep protocol calculates the reputation of a content by collecting neighbor's votes about that content and themselves. Votes about users are employed to weight their votes on contents. Samples of the result set are used to reduce the effect of manipulation. Another mechanism, called Hybrid (COSTA; ALMEIDA, 2007), extends Scrubber to support also votes on contents. Due to its similarity to Credence, Hybrid inherits some of its weaknesses, such as demanding a considerable amount of time to detect polluted contents, especially when attackers behave in coordination (collusion attacks) to circumvent the system and, for example, obtain new identities (whitewashing).

Mislabeled content. User-specified tags have emerged as an alternative, or complementary, way of describing online resources such as web pages, photos, and videos. Since its inception, tags have been largely explored in collaborative platforms such as Delicious, Flickr, and YouTube. In line with their growing popularity, the also called social tagging systems have received great attention from the scientific community. Examples of research directions under investigation include understanding the semantics of tags associated to different types of resources (HECKNER; NEUBAUER; WOLFF, 2008), and the study of their structure and dynamics (MARLOW et al., 2006).

It is consensus in the literature that tags substantially improve searching and retrieving of resources on collaborative platforms, and have the potential to boost reputation systems (MARLOW et al., 2006). Folksonomies add an additional layer of descriptive information about the content, and enable dealing with the subjectivity associated with the process of resource description. In this realm, Andrade et al. have proposed a peer-to-peer annotation approach where users can freely annotate available resources (ANDRADE; SANTOS-NETO; BRASILEIRO, 2008). Despite its potential, one of the main problems is the malicious or accidental assignment of inaccurate/incorrect tags to resources. This behavior, known in the literature as "tag spamming", has been widely discussed in past research (BIAN et al., 2008; KOUTRIKA et al., 2008; BENEVENUTO et al., 2012).

Bian et al. (BIAN et al., 2008) focus their investigation on algorithms to rank social media in question answering portals. In those systems, users can ask for specific in-

formation by posting questions in order to obtain answers from other users. Moreover, users can vote on existing answers to increase/decrease their ranking. Authors describe a machine learning-based ranking framework that integrates user interactions and content relevance to mitigate vote spam attacks. Using a similar strategy, Benevenuto et al. (BENEVENUTO et al., 2012) devise an algorithm to detect video response spam on YouTube. Since users can post videos in response to other users' videos, this may be exploited to increase the popularity of a video or advertise products. Thus, authors present a heuristic that employs attributes of video users and their social behavior to detect spam. Although effective, these solutions require a training data to calibrate the heuristic previously. However, their applicability remains uncertain in systems where such training data is not available or users behave very dynamically.

In another front, Koutrika et al. (KOUTRIKA et al., 2008) carry out an investigation on how tagging systems react to spam. Authors introduce a framework for modeling tagging systems and user tagging behavior. They also describe a method to rank tagged contents based on feedback from users. The proposed method considers the frequency of tags and the reputation of users who assigned them to define content relevance. This approach does not aim at detecting fake content, rather it focuses on giving lower weight to suspicious tag assignments.

General massive attacks

Three most representative massive attacks are presented and discussed in sequence. In *massive lying*, malicious users collude to manipulate voting/reputation mechanisms or subvert distributed protocols. In *eclipse* attacks, malicious users "choke" victims with useless connections to isolate them from the rest of the network. Finally, in *piece corruption* attacks, malicious users replace parts of a content with junk data in order to violate its integrity.

Massive lying. Several protocols and systems present in the literature rely on information provided by users. One of the main algorithms used by the BitTorrent protocol, the Local Rarest First (LRF), considers the frequency in which pieces appear on reports received from users' neighbors. Therefore, massive lying users may provide incorrect information about piece possession and subvert the LRF algorithm (LEHMANN et al., 2012). Lehmann et al. propose a countermeasure that "rotates" the list of neighbors of a user in order to increase the probability of finding potential honest users (LEHMANN et al., 2012). Massive lying attacks also gained attention in online social networks. In this case, users collude to praise their own products or bad-mouth the products of their competitors. Liu, Yang, and Sun (LIU; YANG; SUN, 2008) propose a technique to correlate ratings and users to identify feedback manipulations. They first detect abnormal rating values for individual products. Then, a similarity is calculated among users who provided these ratings. Finally, this similarity is employed to detect and remove ratings from colluding users.

Eclipse. This popular attack in the context of P2P systems is used to isolate victims or split the network (DHUNGEL et al., 2008, 2011). Many malicious users join the system and establish several connections to legitimate users, ultimately degrading the overall QoE. The aforementioned "rotation" algorithm may also be applied to tackle Eclipse attacks (LEHMANN et al., 2012). So and Reeves (SO; REEVES, 2011) propose a solution using locality filters to restrict the neighborhood of a user and mitigate the effect of massive attacks. In another work, Singh et al. (SINGH et al., 2006) design a solution to detect colluding users by monitoring their list of neighbors. This is supported by the assumption

that during eclipse attacks, the number of attacker's neighbors is much higher than average. These solutions require laborious work to combine third-party information and take local decisions.

Piece corruption. This attack consists on malicious users connecting to a victim and uploading junk data (DHUNGEL et al., 2008; YANG et al., 2008). Solutions based on hash functions were proposed to verify the integrity of pieces (HARIDASAN; RENESSE, 2006; MEIER; WATTENHOFER, 2008; SO; REEVES, 2011). In another work, Lehmann et al. (LEHMANN et al., 2012) propose an additional reputation mechanism to isolate users who send corrupt pieces. Although these solutions seem promising, they pose significant costs to either calculate hash functions or build user's reputation.

Conservative strategies

Some practical solutions that use conservative strategies have been proposed to mitigate massive attacks on the Internet. In one front, solutions try to slow down user's actions by using reverse Turing tests, like CAPTCHA (AWERBUCH; SCHEIDELER, 2004). The goal is to impose low costs for individuals and high costs for users who try to abuse the system. In another front, authentication services delay successive requests for identities to avoid brute-force attacks. These strategies not only discourage exhaustive searches, but also protect systems against DoS attacks.

Conservative strategies have been used also to implement anti-spam mechanisms. Li, Pu, and Ahamad (LI; PU; AHAMAD, 2004) propose a framework that slows down spammers by carefully tuning email delivery parameters. The proposed approach adds delay to the email delivery process at the TCP level and increases computation cost at the sender side for delivering emails. In order to slow a TCP connection, authors suggest to postpone TCP acknowledgments or inject congestion notification bits to spoof congestion. Furthermore, to increase sender computation cost, they force the sender to generate more packets (e.g., of a smaller size) to deliver the same message. Results showed that selectively delaying connections can slow down a spammer thousands of times.

Another proposal to fight email spams on the Internet tries to classify email senders into categories such as legitimate, suspicious and bots (HUSNA; PHITHAKKITNUKOON; DANTU, 2008). In this case, bots are considered sources of spamming and must be blocked. The proposed mechanism analyzes traffic from their behavior patterns and applies traffic shaping strategies to delay messages. In order to do that, they analyze the headers of the messages and apply clustering techniques to identify botnet groups. The clustering also considers parameters such as IP address, content length, time of arrival, frequency of messages and content type. Authors show that shaping the transmission channel not only delays spam traffic, but also mitigates automatic use of zombie machines (bots) to send undesirable messages. Moreover, it has been reported that 90% of spam emails can be eliminated by delaying spammer's traffic.

2.3 Summary

CDS are frequently target of malicious interferences. Attackers use content sharing platforms to publish bogus content and/or mislabel existing ones. These attacks are more effective when massively employed by users. One of the main motivations for that is to flood CDS with advertisements/spams. A second motivation is to launch denial-of-service (DoS) attacks and tamper with user's experience, which may be promoted by anti-piracy companies. Content pollution attacks, if neglected, may turn CDS into a useless platform

and compel the users to abandon the system.

As discussed in the previous section, existing solutions use lazy approaches to detect and isolate polluted content. They attempt to identify polluters and/or fake decoys only after evidences are produced, which allows wide dissemination of undesired content until they are definitively banned. Furthermore, many approaches try to derive content relevance based on users' feedback, but they ignore the problem of massive attacks. Colluding users could easily manipulate content relevance by creating sybil identities to provide wrong feedback. In the following chapters, we leverage the power of conservative strategies to mitigate massive attacks in CDS.

3 CONSERVATIVE APPROACH BASED ON BINARY VOTES

Considering the limitations of the aforementioned solutions, this chapter introduces a conservative strategy, and corresponding mechanism, to control content pollution dissemination in CDS. The strategy counts positive and negative votes assigned to contents in order to classify them as either non-polluted or polluted. The mechanism, called FUNNEL, operates by controlling the distribution of copies according to votes assigned by community users. That is, the number of concurrent downloads permitted is affected by the proportion between positive and negative votes. An incentive mechanism elicits feedback from users about downloaded contents.

FUNNEL was initially designed to target closed BitTorrent communities, i.e., communities that require user registration before publishing and/or retrieving torrents. Although FUNNEL was instantiated in a controlled environment, it may also be easily deployed in real world communities, since its design does not require any modifications to BitTorrent agents. An implementation of the mechanism was evaluated through experiments. Results indicate that FUNNEL prevents polluted copies from being disseminated, whilst imposing a low overhead in the distribution of non-polluted ones.

Organization. The remainder of this chapter is organized as follows. Section 3.1 describes the proposed strategy and the mechanism that instantiates it, coined FUNNEL. Section 3.2 presents the results of an experimental evaluation performed with FUNNEL to analyze the technical feasibility of the approach. Section 3.3 enumerates a list of considerations on the proposed solution. Finally, Section 3.4 closes the chapter with a summary.

3.1 FUNNEL model

This section describes the proposed strategy to fight content pollution and details FUNNEL – the mechanism instantiation for BitTorrent file sharing communities. Our key observation is that soon after some new content is published, there is no evidence whether it is legitimate. The legitimacy of the content can only be established after a substantial number of users have downloaded and evaluated the content, as well as provided feedback about it. This may take some time, and some users may even refuse to provide feedback. If no control is in place, a polluted content could be widely disseminated before it is identified as such. Previous approaches may demand a large number of negative evaluations before they are able to classify a given content as polluted.

3.1.1 Overall strategy

The distinguishing aspect of our strategy is to start with a very low dissemination rate, growing quickly as long as positive feedback (towards the legitimacy of the content)

is provided by users. For the sake of clarity, we describe the strategy considering the dissemination of a single content version, since the strategy is independently applied to every content version published.

A binary voting mechanism is used to build the reputation of a given object in the community (R). When a user finishes retrieving the content, he/she issues a positive vote to testify the legitimacy of the downloaded version, or a negative vote to report the version as polluted. The content reputation is supported by the Subjective Logic (JOSANG; POPE; HAYWARD, 2006) and is computed using the fraction of positive votes out of the total number of votes issued to a given version. The reputation, which is defined in the interval $[0; 1]$, is expressed by Equation 3.1. Variables p and n represent, respectively, the number of positive and negative votes issued by users. In addition to the number of votes, the equation includes a constant factor a that defines the behavior when few votes are available: if $p, n \to 0$, then $R \to a$.

$$R = \frac{p + 2a}{p + n + 2} \tag{3.1}$$

Since the strategy relies on votes, it could be thwarted by malicious users that do not download the content and/or vote multiple times. Further, the strategy requires that most non-malicious users vote and do so truthfully. These issues will be addressed in Sections 3.1.4 and 3.1.5, respectively.

The value of R is used to establish the number of concurrent downloads allowed in a given instant, denoted as A. The value of A is calculated according to Equation 3.2, as a function of the corresponding content reputation (R), and assumes a value in the discrete interval $[A_{min}, A_{max}]$ (with $A_{min}, A_{max} \in \mathbb{N}^*$). In this context, A_{min} denotes a lower bound, which guarantees a minimum number of downloads (when $R \to 0$, $A \to A_{min}$).

As soon as the perceived content reputation surpasses the reputation threshold r (i.e., when $R \geq r$), downloads are not restricted and peers are free to download. Otherwise, content dissemination is ruled by the current value of A, being limited by A_{max} – which denotes an upper bound for the number of downloads to be allowed by the strategy (when $R \to 1$, $A \to A_{max}$).

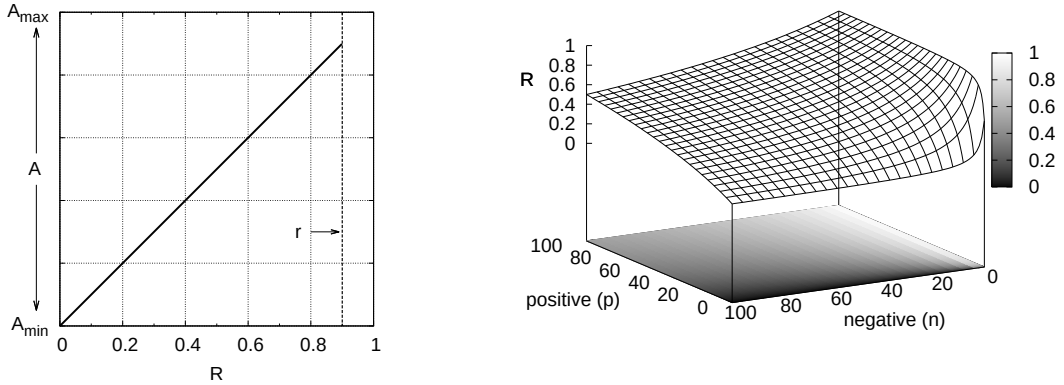
$$A = \begin{cases} \infty & \text{if } R \geq r, \\ R \times (A_{max} - A_{min}) + A_{min} & \text{otherwise.} \end{cases} \tag{3.2}$$

Three important properties hold for Equation 3.2:

- **Property 1:** For any $r \in (0, 1]$ and $R \geq r$, $A = \infty$; when so, the content will be assumed beyond suspicion and unlimited downloads will be allowed;
- **Property 2:** For any $r \in (0, 1]$ and $R < r$, $A = R \times (A_{max} - A_{min}) + A_{min}$, situation in which content dissemination will be ruled according to its perceived reputation;
- **Property 3:** When $r = 0$, $A = \infty$ regardless of R; when so, no control will be imposed and peers will be free to download the content, regardless of the perceived reputation.

Figure 3.1(a) illustrates the behavior of variable A as a function of the reputation score R. For instance, when $R \geq r = 0.9$, an arbitrary number of downloads can take place; if new negative votes are provided, the reputation score could fall below 0.9, with the

limitation being reinstated. Figure 3.1(b) evaluates the variation of reputation according to Equation 3.1, as a function of the (positive and negative) votes issued to it.



(a) Relationship between reputation and concurrent downloads

(b) Reputation behavior in terms of assigned votes ($a = 0.5$)

Figure 3.1: When content reputation R exceeds r, it is deemed non-polluted (a), which occurs for a specific vote range (b)

The number of concurrent downloads actually taking place at any moment is denoted as D. The value of D is incremented when a download begins (or resumes), and is decremented when it ends (or is interrupted). Although $D \leq A$ is desirable, since both A and D fluctuate, it is possible that A drops and temporarily $D > A$. To establish if a new download is possible, the values of A and D are compared. If $D < A$ (or $R \geq r$), new downloads can be allowed.

In the following subsections, we describe how this strategy was designed and instantiated in BitTorrent file sharing communities. With FUNNEL, we initially target closed communities, in which users obtain a weak identity (an email address needs to be provided and confirmed) and use it for authentication purposes before accessing the portal.

The remainder of the section is organized as follows. The method applied to estimate the current number of concurrent downloads (D) in a BitTorrent swarm is described next in Subsection 3.1.2. The technique employed by the tracker to control the number of concurrent downloads in the swarm is then presented in Subsection 3.1.3. In Subsection 3.1.4 we discuss how to allow only a single vote per community user. The presentation of the mechanism is concluded in Subsection 3.1.5, with the description of the voting incentive mechanism.

3.1.2 Estimating the number of concurrent downloads

To apply the dissemination control mechanism, the tracker needs to calculate the number of peers simultaneously retrieving a given torrent (D). Each torrent may be regarded as a version of a published content. Participating peers act autonomously and without the trackers' intervention, hampering a precise estimation of D. Therefore, when instantiating the strategy for BitTorrent, the value of D will be approximately the number of leechers in the swarm. To determine D, the tracker keeps track of which peers are in the swarm (peers are removed after some time of inactivity) and their progress in terms of download and upload. The estimation of D is based on the number of online peers with incomplete downloads.

Since there is no guarantee about the download progress information sent to the tracker, malicious peers might pretend they are still downloading the torrent. Hence, a massive attack could lead the tracker to overestimate the value of D, potentially causing new download requests to be denied (new downloads are allowed when $D < A$). In fact, malicious peers might remain in the swarm with little or no participation, contributing just enough to occupy the present slots and prevent new downloads from starting. However, in a content pollution attack, the objective of the polluter is the opposite: to increase the number of concurrent downloads.

Nonetheless, malicious peers might aim at causing a denial-of-service (DoS) attack against peers sharing a non-polluted content. There is a cost associated with this attack, given that malicious peers must continuously announce themselves to the tracker in order to be seen as leechers. To overcome this limitation, a probabilistic strategy could be employed, in which requests that would be denied are granted according to some probability. Honest peers would eventually get a chance to proceed, preventing malicious peers to take over the swarm. Lehmann et al. (LEHMANN et al., 2012) propose a "rotation algorithm" to protect swarms against similar massive attacks. Another possibility would be to enforce an upper bound on download times (based on file size or the average of observed download times). While it would be relevant to quantify the effect of both strategies, we leave their exploration for future work.

3.1.3 Adjusting the number of concurrent downloads

In order to control content dissemination in the community, the tracker can only resort to the list of peers provided to BitTorrent agents. The strategy adopted by FUNNEL consists in adjusting the *size of this list* depending on whether the request for download should be granted or not. In the former case, the tracker acts according to its normal behavior, and provides the agent with a regular list. Otherwise, an empty list is sent to the peer[1]. A session starts when a peer P receives a non-empty list (the tracker has given authorization). As mentioned earlier, P continues receiving lists even if $D \geq A$ becomes true (because the value of A has fallen). The value of A may be decreased reflecting a drop in reputation R. However, if the session with P expires, a future request of P for download will be evaluated again by the tracker.

It is also important to consider the situation in which a peer P has its request denied, but attempts to bypass the control and join the swarm. Current BitTorrent agent implementations typically include two extensions which are employed in the discovery of peers: *PeerExchange* (PEX) and Distributed Hash Tables (DHTs). Since it is a gossip-based protocol, PEX does not affect FUNNEL: new peers can only be discovered when *at least* one peer is previously known. If P is attempting to join the swarm for the first time, it is not aware of other peer addresses.

In contrast, the use of DHT enables the discovery of peers without needing a tracker, which would allow peers to bypass FUNNEL. Torrent files may contain the address of one or more nodes of the DHT; if reached, these nodes could provide IP addresses of peers in the swarm. To assess the real impact of this limitation, we measured the popularity of DHTs in several popular BitTorrent communities. To this end, thousands of torrent files were collected from six communities and analyzed. Figure 3.2 summarizes the number of collected files and the fraction of DHT-enabled torrents in each community. The small fractions correspond to contents that might be retrieved by peers in the

[1]We did not explore in this thesis the possibility of sending partial lists.

swarm without a tracker, and therefore circumventing FUNNEL. This low adoption may be due to several practical limitations of DHTs, including their susceptibility to malicious behavior (URDANETA; PIERRE; STEEN, 2011; TIMPANARO et al., 2011). However, a community wishing to enforce content pollution control might discourage or forbid its users from publishing DHT-enabled torrents (this seems to be common-practice already in some closed communities).

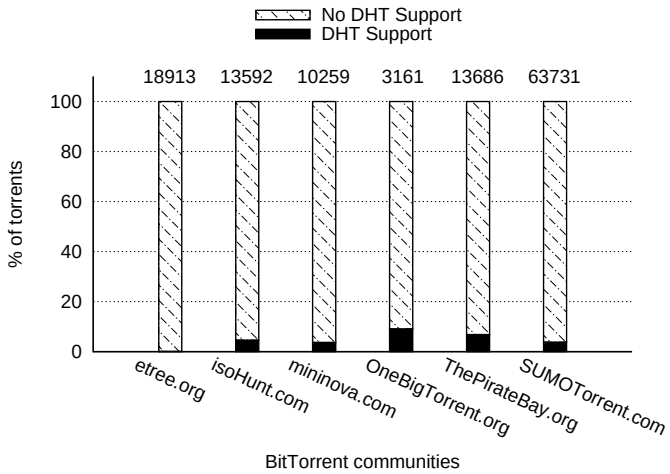


Figure 3.2: Measurements on DHT support across BitTorrent communities

3.1.4 Ensuring unique votes per user

Recall that FUNNEL targets the dissemination of polluted content in closed communities. Hence, each interaction of a peer with the tracker can be associated to the corresponding user identity in the portal. The portal aggregates the votes issued by users to each of the published torrents, storing tuples in the format $\langle Peer, Torrent, Vote \rangle$. The portal also uses this information to count the number of positive and negative votes assigned to each torrent.

Note that a peer may be able to identify a polluted version even before the download is completed. Therefore, as soon as the peer has enough evidence that the version is polluted (e.g., by *previewing* part of audio/video contents), it may halt the download and provide its (negative) feedback.

A set of malicious peers may take advantage of this characteristic to perform a collusion attack. Malicious peers could try to manipulate the reputation of a torrent, by voting (negatively) against non-polluted contents, and (positively) for polluted ones. In both cases, the reputation might affect the dissemination of a content contrary to what was intended by normal users. To mitigate the impact of such an attack, the portal keeps a record indicating in which torrents users have participated, even though temporarily, and accepts votes provided by these users only. In addition, resources such as computing challenges (e.g., CAPTCHAs) may be employed to mitigate this kind of attack (AWERBUCH; SCHEIDELER, 2004).

3.1.5 Incentives for users to vote

The proposed mechanism uses the votes issued by users to build up the reputation of torrents in the community. If users retrieve a content but often fail to provide a feedback on its legitimacy, the mechanism becomes ineffective. To address this issue, an incentive

mechanism is required. Two approaches were considered: reward users who vote, and punish those that do not. Rewarding users who vote might induce large amounts of indiscriminate votes, since users would attempt to increase the amount of benefits. Punishing users who do not vote, conversely, is susceptible to *whitewashing* (FELDMAN et al., 2006), although there is a non-negligible cost to obtain a new identity.

FUNNEL follows the second approach: the tracker employs a policy to "punish" users who do not vote. Similarly to the approach presented in Subsection 3.1.3, the tracker sends peer lists whose sizes are affected by the fraction of torrents with user's votes out of the total torrents in which this user has participated. Therefore, let N represent the number of peers in the original list that would be provided by the tracker, V the number of torrents in which the user has voted, and P the number of torrents in which he/she has participated, the size S of the list provided to a requesting peer is computed as in Equation 3.3.

$$S = N \times \min\left\{\frac{V+1}{P}, 1\right\} \tag{3.3}$$

Given that a user cannot vote about a torrent without having joined the corresponding swarm, and that votes may be issued before the download completion, it follows that $V \leq P$. The equation tolerates one missing vote without decreasing the peer list size, i.e., if the difference between V and P is greater than 1, the fraction will be lower than 1. The operator $\min$ guarantees that the size of the list does not exceed N, which happens when votes are issued prior to the complete retrieval of a content ($V = P$).

3.2 Evaluation

This section presents an experimental evaluation of FUNNEL. The purpose of this evaluation is to provide answers to three fundamental questions, attempting to quantify: (i) the negative impact caused by the mechanism in a scenario where a non-polluted content is disseminated or, in other words, the overhead imposed by FUNNEL with non-polluted contents; (ii) the effectiveness of FUNNEL in slowing down the dissemination of polluted contents to normal users; (iii) the negative impact of collusion attacks performed by malicious peers against the mechanism.

Each of the scenarios employed in the experimental evaluation comprised the dissemination of a single content version (one torrent), instead of many, due to two reasons. First, unlike other approaches, FUNNEL spawns individually and independently to every content version. Second, it is very hard, if not impossible, to safely model user behavior when facing multiple torrents of the same content. Users may decide according to factors such as the publisher identity, the current number of seeders, of leechers, and number of torrents downloaded. Hence, although FUNNEL may influence the user's choice of torrent by decreasing the number of leechers (and indirectly, of seeders), the choice will be affected by multiple subjective factors. We are not aware of any study that analyzes the rationale behind this choice and reliably establishes a user behavior pattern.

In Subsection 3.2.1, we introduce the experimental setup and the notations employed. Results are, then, discussed in three parts. First, in Subsection 3.2.2, we show the effectiveness of the solution when a non-polluted and a polluted content are published. Second, in Subsection 3.2.3, we evaluate the strategy that controls the number of concurrent downloads taking place. Third, in Subsection 3.2.4, we present the evolution in the proportion of honest and malicious users who had access to the content.

3.2.1 Experiment details

Two very different scenarios were evaluated: honest peers seeding a non-polluted torrent, and malicious peers seeding a polluted torrent. In the former case, an attacker will try to prevent the distribution of the content, whereas in the latter the attacker will attempt to have the content disseminated among honest leechers. We refer to these scenarios as 'non-polluted' and 'polluted', respectively.

Let I denote the number of initial seeders (either honest or malicious) that start a swarm and remain available during the entire experiment. Let C denote the total number of honest/correct peers that will arrive. The number of malicious peers who attack the system represents a proportion M of the honest peers.

We examined (both through simulation and experimentally) a range of values for I, C, and M to assess parameter sensitivity. Not only the scale of peers has been varied (from hundreds to thousands of peers), but also the proportion among the values assigned to I, C, and M. Since the results observed were consistent for all combinations of parameters, we chose to describe (and present the results obtained for) an intermediate-size scenario, with $I = 20$ and $C = 500$. To assure that these choices were representative, we monitored 30 popular trackers and examined 3,167,832 torrents. Figure 3.3 shows the cumulative distribution function (CDF) of swarm sizes and the distribution of users among these swarms. We notice that swarms with 500 peers or less correspond to 99.94% of all torrents and include 80% of all peers.

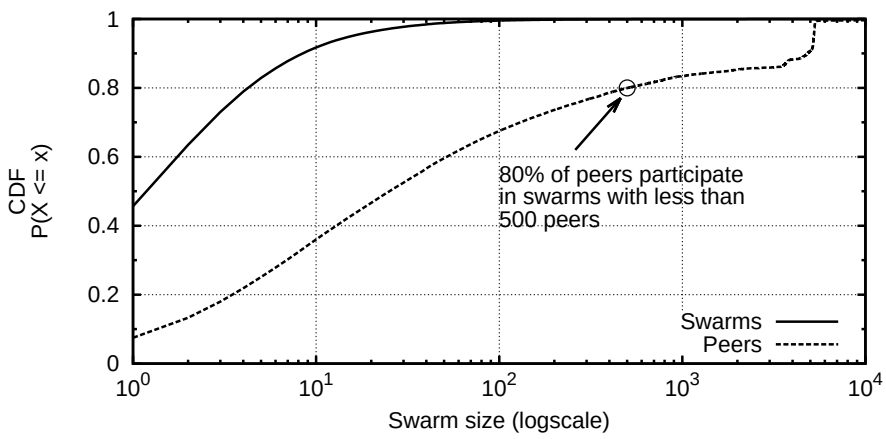


Figure 3.3: CDF of swarm sizes and distribution of users among these swarms

The proposed mechanism employs a voting-based reputation scheme to classify contents as being polluted or non-polluted. Such schemes are subject to collusion attacks, in which a malicious user lies to manipulate the reputation of a given content. Hence, in both polluted and non-polluted scenarios, the attacker leverages a proportion of M colluding peers to attempt to defeat the mechanism. They will not only redistribute polluted content, but also provide false feedback. In the context of this work, an attacker votes positively to a polluted content and negatively to a non-polluted one. For the experimental setup being presented, the value of M was varied from 0 to 30% (note that these malicious users will need to have an account in the community).

First, the swarm is created with I initial seeders. To evaluate the most damaging scenario, malicious peers join immediately afterwards and contact the tracker. Honest peers, in turn, join the swarm following an exponential distribution, in line with measurements presented in (GUO et al., 2005). Peers leave the swarm only after completing their down-

loads, obeying a *minimal contribution metric*, denoted as ρ. The value of ρ represents the amount of data a peer is willing to upload in regard to what he/she has downloaded. For example, if $\rho = 0$, a peer will leave the swarm right after completing the download, regardless of the amount he/she has uploaded. If $\rho = 2$, it means a peer will attempt to upload at least twice as much content as he/she downloaded. Note that a user relies on interested leechers in order to upload content until its ρ is fulfilled. Conversely, a user might upload substantially more than what its ρ would indicate, since ρ is in effect only after the leecher becomes a seeder.

In this context, we define the behavior of honest and malicious peers when retrieving polluted and non-polluted contents. There are four possible cases, as follows. When an honest peer completes the download of a polluted content, we assume that it immediately detects the pollution and therefore leaves the system ($\rho = 0$). Likewise, if a malicious peer finishes retrieving a non-polluted content, it does not disseminate it ($\rho = 0$). Although the idea of leaving the system immediately does not mimic real user behavior, we left this factor out of the evaluation in order to simplify the analysis.

In contrast, when a malicious peer retrieves a polluted content, it remains in the swarm contributing indefinitely to its dissemination ($\rho = \infty$). The fourth and last case corresponds to the 'normal case': an honest peer retrieves a non-polluted content. The level of collaboration in BitTorrent communities have been up to debate, so it is hard to assign 'real' values of ρ to peers. Adopting a methodology in line with a previous study (ANAGNOSTAKIS et al., 2006), we assume that after (fully) downloading the content, 25% of leechers leave the swarm without becoming seeders ($\rho = 0$), 41% are willing to upload as much as they have downloaded ($\rho = 1$), and 34% attempt to upload twice as much as they have downloaded ($\rho = 2$).

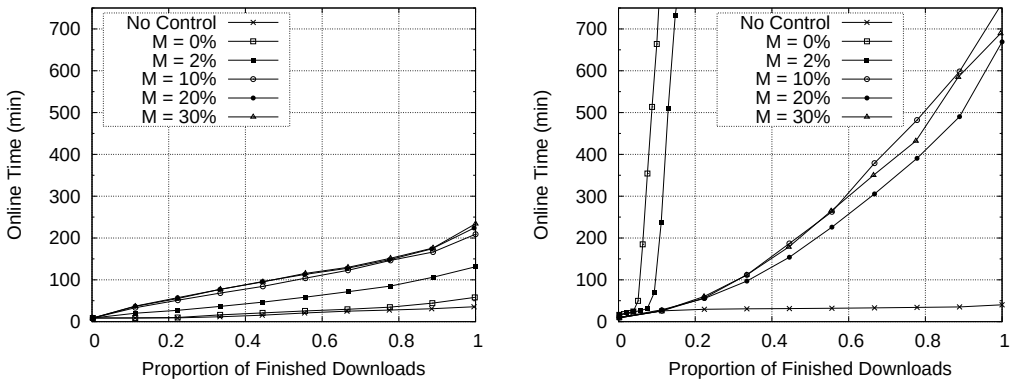
To keep the campaign of experiments manageable, we chose relatively small torrents: 60 MB. The set of BitTorrent agents – the standard implementation of BitTornado – was distributed among 10 dedicated machines, interconnected through a 100 Mbps network switch. Agent configuration included a limit of 7 upload connections and infinite number of download connections (in larger settings, the operating system would have forced a limit on this value). In addition, we adjusted the upload and download rates of the agents to 256 Kbps and 1 Mbps, respectively. Finally, we have assigned the following values for FUNNEL input parameters: $A_{min} = 1$, $A_{max} = 50$, $a = 0.5$, and $r = 0.95$. These values were chosen because they represent a reasonable balance between a more conservative, rigid dissemination policy and a more relaxed one.

3.2.2 Effectiveness of the mechanism

To measure the effectiveness in both polluted and non-polluted scenarios, we measured how long it took to each honest peer to complete its download. Figure 3.4 illustrates the amount of time peers remain online (up to completion) in both non-polluted and polluted scenarios (Figure 3.4(a) and Figure 3.4(b), respectively). A pair *(x,y)* in the plot means that a fraction of peers *x* remains online for up to *y* minutes in order to complete their download.

Figure 3.4(a) illustrates the behavior of the mechanism for non-polluted scenario. In this case, the lower the curves, the better. First, note that the curves 'No Control' and '$M = 0\%$' are similar. That is, when there are no attackers, the presence of the pollution control mechanism introduces a negligible overhead. More precisely, results indicate that the delays imposed by the mechanism are not larger than 10 minutes to 80% of peers.

When running an experiment with 2% of malicious peers, the mechanism receives



(a) Non-polluted content (b) Polluted content

Figure 3.4: Effectiveness against pollution attacks with colluding peers

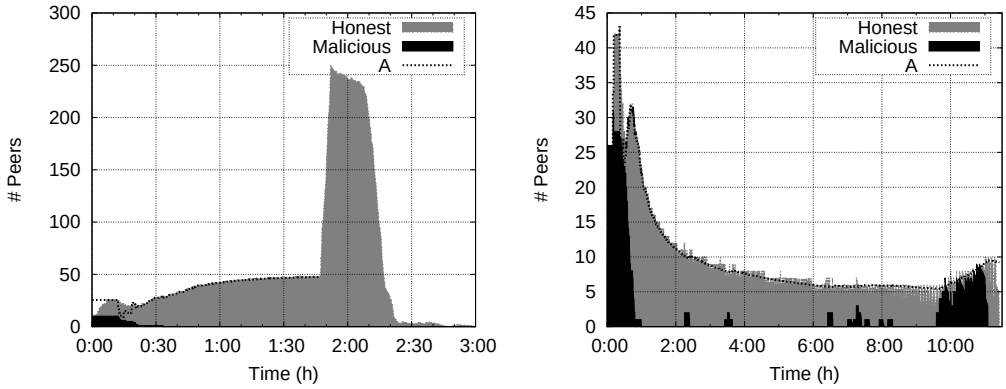
votes against the legitimacy of the content (negative), which induces FUNNEL to slow down the dissemination of non-polluted content. This may be observed by the fact that, in comparison to the curve '$M = 0\%$', honest peers took at most 70 extra minutes to download the content. However, we note that, to duplicate this delay, it takes five times more attackers ($M = 10\%$). In addition, the efficiency of the attack decreases as the proportion of malicious peers exceeds 10%. Such phenomenon occurs because when colluding peers arrive, at most $A = [0 + (1 - 0) \times 0.5] \times (1 + 50) + 1 = 26.5$ of the malicious peers are allowed to start their downloads ($R = 0.5$ since in the initial stage there are no votes recorded, thus $R = a$). Therefore, only the remaining malicious peers may "compete" with the honest peers in the swarm for an opportunity to start the download.

Results regarding the polluted scenario are shown in Figure 3.4(b). In the absence of control, 100% of peers complete the download of the polluted content in less than 50 minutes. In this case, the benefits of using FUNNEL are *highly expressive*: approximately 90% of peers consume more than 750 minutes (12.5 hours) to complete the download of the polluted content (curve '$M = 0\%$'). For the sake of legibility, the plots in the figure were limited in the *y*-axis. Please note, however, that the overall behavior of curves $M = 0\%$ and $M = 2\%$ remained unchanged, even past 15 hours of experiment. The mechanism also performs efficiently, controlling the dissemination of polluted content, even in the presence of 2% of malicious peers voting positively. When $M \geq 10\%$, the set of malicious users obtained success in promoting content dissemination; however, note that to obtain the 60 MB file, it took the honest peers approximately 600 minutes (10 hours), 12 times more than in the absence of FUNNEL. We emphasize that, in the experiments carried out to evaluate our proposal, peers do not give up on trying to obtain a non-polluted content. In practice, peers may leave the network due to the delay to start the download, so that FUNNEL may be actually more efficient in holding pollution than shown in the plot.

3.2.3 Setting the number of concurrent downloads

In an in-depth analysis of the evaluated scenarios, it is possible to precisely observe the performance of our mechanism in controlling the dissemination of polluted content among peers belonging to the swarm. Figure 3.5 presents how the number of peers (honest and malicious) evolves in the observed swarms. The plot shown in Figure 3.5(a) illustrates an environment in which the dissemination of a non-polluted content is hampered by 10

malicious, colluding peers (2%). In the first 10 minutes, the swarm is taken over by these attackers, being followed by $A - 10 = 16$ honest peers. When the malicious peers issue their negative vote, the value of A, computed by the tracker and represented in the dotted line in the plot, is decreased. The attack leads to a reduction in the number of peers retrieving the content (peers currently in the swarm complete their downloads and new peers are prevented from joining because of the lower value of A). However, the positive votes (issued by the honest peers) cause an increase in the value of A, which reaches A_{max} after approximately 1 hour and 45 minutes of experiment. Such effect may be observed in Figure 3.5(a), with the end of the dotted line and the surge in the number of honest peers.



(a) Non-polluted content ($M = 2\%$)

(b) Polluted content ($M = 20\%$)

Figure 3.5: Number of downloads during the dissemination of a content

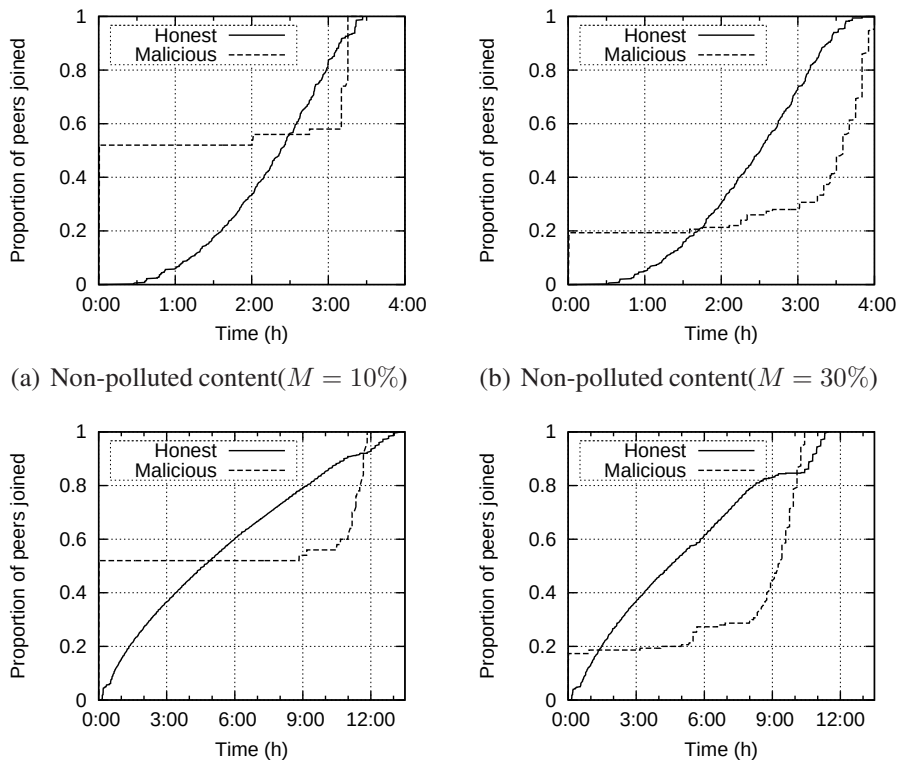
The plot in Figure 3.5(b) illustrates an environment in which the dissemination of a polluted content is supported by 100 malicious, colluding peers (20%). In this case, malicious peers completely take over the system, impeding honest peers to retrieve the content in the initial stage. However, FUNNEL prevents $100 - A = 74$ attackers to start their downloads, and these must "compete" for an opportunity along with honest peers that arrive in the swarm. Observing Figure 3.5(b), one may note the effect caused by the attacker's positive votes: an increase in the value of A occurs during the first 30 minutes of the experiment. As new downloads are authorized, both honest and malicious peers enter the swarm. The votes of honest peers cause a decrease in the value of A, compensated by a minority of counterfeit votes provided by attackers which had the opportunity to begin their downloads. This procedure occurs along the experiment and allows us to conclude that the system in fact adapts and reacts quickly, depending on the number of attackers, towards a stable situation in which non-polluted versions are widely disseminated and polluted ones are slowly disseminated among peers.

3.2.4 Impact of the mechanism on peer joins

In this subsection we analyze the effectiveness of FUNNEL from a different perspective. Instead of focusing on the number of completed downloads, we now concentrate on the number of honest and malicious peers starting their downloads. Since FUNNEL employs a conservative strategy, we say that peers *arrive* when they first attempt to start their downloads, and actually *join* when they are allowed to start. In this analysis, we observe the proportion of peers (honest and malicious) during the distribution of a polluted and non-polluted content. Both scenarios were instantiated with 10 and 30% of attackers, and a fixed number of 500 honest peers.

Figure 3.6 shows the cumulative proportion of peers during the entire experiment. A pair (x, y) in the plots means that a proportion of y peers (from all honest or malicious, depending on the curve) had access to the content by instant x. When the 'Honest' curve reaches the top of the figure, it means all 500 honest peers were allowed to join the system. Similarly, when the 'Malicious' curve reaches the top, all attackers were allowed in.

Since malicious peers arrive first, they get hold of the first slots to download the content. During the experiment, honest peers arrive and compete as well for slots of download, thus increasing the set of waiting peers. Both honest and malicious peers gradually access the content and cast their votes. In the non-polluted scenarios, attackers try to join and vote in order to deny the access of other peers to the system. In the polluted scenarios, they try to ease the access of other peers.



(a) Non-polluted content($M = 10\%$)

(b) Non-polluted content($M = 30\%$)

(c) Polluted content ($M = 10\%$)

(d) Polluted content ($M = 30\%$)

Figure 3.6: Proportion of peers (honest and malicious) joining the system during the dissemination of a content when FUNNEL is present

Figures 3.6(a) and 3.6(b) represent the non-polluted scenarios, for $M = 10\%$ and $M = 30\%$, respectively. First note that, by definition, curves are proportional and will reach 1, but in different time scales (3h-13h). These figures illustrate the benefits of the control mechanism to honest peers. This is due to the fact that the conservative mechanism holds together malicious and honest peers, increasing the proportion of honest peers over time. This leads to a scenario in which the majority of peers waiting for access are honest and therefore the probability of a honest peer joining the system becomes higher. Recall that peers are induced to wait by receiving empty lists. If the user does not want to wait, he might look for another potentially good content. For $M = 10\%$ (Figure 3.6(a)), at time 3 hours and 10 minutes, we notice that more than 90% of honest peers have managed to join the system, whereas only 55% of attackers have had access. At the same time, for $M = 30\%$ (Figure 3.6(b)), the proportion of attackers remains close to 30%, and

the waiting attackers finally join when more than 80% of honest peers have accessed the system. In the absence of FUNNEL, in contrast, 100% of attackers would instantly join the system.

In the polluted scenarios (Figures 3.6(c) and 3.6(d)), FUNNEL provides a similar benefit to honest peers. They join the system, download the polluted content, and vote negatively about it. As the number of allowed downloads decreases, due to honest votes, attackers face an even lower probability of joining the system. This explains why only 50 and 30% of attackers, for $M = 10\%$ and $M = 30\%$ respectively, join the system during the first 8 hours of experiment, 5 hours later if compared to the non-polluted scenarios. When the curves of malicious peers start growing (about 8 hours in the figures), almost 80% of honest peers have accessed the system.

In summary, FUNNEL reduces the proportion of malicious peers accessing both non-polluted and polluted contents during the initial phase of its dissemination, when most honest peers arrive. This positive effect is due to the contention strategy of our mechanism: since honest and malicious peers wait until joining the system, collusion and massive attacks are "diluted" and fewer malicious votes are casted over time. Consequently, the potential harm that an "*en masse*" arrival of malicious peers could cause is reduced.

3.3 Considerations on the proposed solution

As presented in the previous sections, FUNNEL enables the quick dissemination of legitimate versions of a content, while limiting the spread of polluted ones, even in the presence of malicious, colluding peers. An aspect of paramount importance concerning our solution is its relevance and impact for BitTorrent users: there is a significant number of communities (therefore, users) that might take advantage of FUNNEL, as presented in this chapter. Nonetheless, our solution has the potential to be extended to a broader, wider range of scenarios (e.g., to enable its applicability to open communities and other P2P file sharing systems). A number of issues must be addressed in order to accomplish such a generalization, for example (*i*) dealing with the presence of counterfeit identities (sybil nodes) (DOUCEUR, 2002), (*ii*) aggregating positive and negative votes, and (*iii*) computing the number of concurrent downloads, (being the latter two more relevant in the case of purely decentralized P2P systems).

FUNNEL relies on the feedback provided by users to compute the reputation of a given content, and adjusts the number of concurrent downloads according to the resulting reputation. Since our solution controls the dissemination of a given content, we decided not to consider an extra mechanism to assign scores to users. However, additional strategies to create users' reputation and moderate their activities can be designed as a complement to the current solution. In this case, votes assigned by users might be weighted by their reputation and, hence, FUNNEL would potentially converge faster. However, even with user incentives to elicit feedback, some users may remain uncooperative (i.e., unwilling to vote). A lower number of votes tends to delay convergence; in the meantime, the mechanism will conservatively allow a fraction of downloads to take place. Like other P2P systems, our approach relies on the cooperation of a fraction of users, who will either respond to the incentive mechanism or simply behave altruistically.

In regard to the dissemination control, malicious peers might launch an attack against users sharing non-polluted contents by taking over all the download slots when only A_{min} are available (e.g., after a previous attack to decrease the content's reputation). The A_{min} slots would then be occupied by malicious peers pretending to have limited bandwidth

capabilities (e.g., 1 Kbps). Consequently, legitimate users would not be able to download the content and regenerate it (providing positive votes to raise its reputation). A simple approach to overcome this threat would be setting expiration times for the download authorizations. While this is an intuitively effective strategy, the optimum value for this limit is not obvious. If this period is too short, honest peers would prematurely loose the slot. This means that they would be able to keep on downloading, although would not receive fresh information about new peers. Conversely, if the period is too long, malicious peers would be able to hold their slots for longer times, and consequently cause a larger impact on the system. While it would be relevant to elaborate on how to optimally set this period in a dynamic context, this is out of the scope of this thesis.

Finally, we expect objects with lower popularity to be more susceptible to distributed denial-of-service attacks. If there is not much interest on a content being shared in the network, fewer resources might be necessary for an attacker to take control over it (either by providing false testimonies or by requesting, but not finishing, several downloads in parallel). Nevertheless, considering the exponential distribution of users among contents, a reduced number of users would be affected by attacks on objects of low popularity. Substantial resources would be needed to harm a reasonable amount of users. While this might represent indeed an attack scenario, it is of smaller relevance. Instead, we have concentrated on cases where larger number of users might be affected by a single attack.

3.4 Summary

BitTorrent communities have become one of the most popular alternatives for file sharing. One of the side effects of such a high popularity is that they have also become a natural target of content pollution attacks, where malicious users disseminate corrupted copies, viruses, or other malware. In addition to contributing to the spread of malicious decoys, content pollution may also promote the waste of client and network resources, considering that peers may attempt to obtain titles through repeated downloads. To mitigate this problem we proposed a novel content pollution control strategy and corresponding mechanism, FUNNEL, to reduce the dissemination of polluted content according to users' perception on the content's legitimacy. FUNNEL addresses the major shortcoming of manual inspection required by traditional, *ad hoc* proposals, by automatically adjusting the number of concurrent downloads according to the proportion between positive and negative votes.

We have shown the effectiveness and technical feasibility of the proposed mechanism through an experimental evaluation, carried out using an implementation of FUNNEL. The results obtained – considering real BitTorrent agents on live networks – also demonstrate the efficiency of the proposed solution, which has imposed significant delays to the dissemination of polluted contents, whereas causing a relatively low delay for the dissemination of non-polluted, legitimate ones. Furthermore, low dissemination delays experienced by non-polluted copies have been observed even when there is a large number of peers colluding to hamper the dissemination. These observations show that our conservative approach not only controls the dissemination of polluted content, but also requires a significant effort to adversely manipulate the mechanism.

Although FUNNEL achieved satisfactory results, it relies on a binary voting schema, which polarizes contents in either polluted or not, and is unable to deal with the inherent *subjectivity* to characterize "pollution". This characteristic assumes that users have similar opinions about contents and are able to accordingly detect inappropriate metadata,

which was already shown to be false (LEE et al., 2006). As a next iteration to tackle the pollution problem, a more flexible mechanism is designed in order to enable users to express their opinions beyond simply positive and negative votes. The next chapter presents our solution that aims at providing users a better quality of experience in CDS.

4 EXTENDING THE MODEL TO DEAL WITH SUBJECTIVENESS

As an important step towards increasing download quality in content distribution systems, this chapter presents DÉGRADÉ, a novel tag-based strategy to control the dissemination of undesired contents. Unlike existing solutions, DÉGRADÉ leverages social tagging system techniques to allow users to express their perception about downloaded contents in a flexible and accurate fashion. Furthermore, and key to our solution, it restricts the dissemination of contents when the uncertainty on the vocabulary describing them is high, and promotes their dissemination otherwise. DÉGRADÉ goes beyond the traditional binary polarization ("polluted" *vs.* "non-polluted") of users' feedback in characterizing contents and provides a more effective way for users to express their "taste". The control strategy underneath manages to reconcile such "tastes" and downloads.

DÉGRADÉ has been evaluated by means of a large set of simulations. BitTorrent swarms have been reproduced to mimic various scenarios of content dissemination, and real traces from Delicious have been employed as sequences of tag assignments. In addition, attacks have been designed to induce users to download undesired contents, and their impact to the proposed strategy have been analyzed. The results achieved show that our strategy is capable of significantly improving the "quality" of downloads even in the presence of malicious behaviors.

Organization. The remainder of this chapter is organized as follows. Section 4.1 presents the proposed strategy and equations that govern its behavior. Section 4.2 discusses evaluation results. Section 4.3 enumerates a list of considerations on the proposed solution. Finally, Section 4.4 closes the chapter with a summary.

4.1 DÉGRADÉ model

This section introduces the downloads control strategy supported by content vocabulary variation. This strategy, called DÉGRADÉ, can be divided in two main components. The first one is the *tagging component* used by users to type in the set of tags associated to some content. The second one is the *aggregator* service, where the tags are stored and employed to build our solution. The tagging component allows users to annotate contents similarly to the popular Delicious bookmarking system. After each tag assignment, the aggregator service updates the content vocabulary and the variation metric used to control further requests for download. In the next subsections, we detail how DÉGRADÉ fundamentally works. First, in Subsection 4.1.1, we investigate the typical dynamics observed in vocabularies of collaborative tagging systems. Second, we define a metric that captures such vocabulary variation, in Subsection 4.1.2. Third, in Subsection 4.1.3, we elaborate

on the key idea behind DÉGRADÉ: as the variation increases (and therefore the metric value), less downloads are allowed to proceed.

4.1.1 Stable patterns in tag proportions

In the proposed strategy to fight pollution, users describe contents using tags. Given a set $\mathcal{T}$ of n possible tags to assign to any content, we define a tag assignment through a vector of n binary elements indexed by those tags. So, for $|\mathcal{T}| = n$, the i-th assignment is represented as a vector $\vec{v}_i = \langle v_{i,t_1}, \ldots, v_{i,t_n} \rangle$, whose element v_{i,t_j} is set to 1 if the assignment contains a tag t_j, and 0 otherwise. A sequence of assignments defines the content vocabulary, which represents the entire set of tag frequencies. Those frequencies are called proportions when normalized by the total amount of tags. Consider the matrix M with m assignments:

$$M_{m \times n} = \begin{pmatrix} \vec{v}_1 \\ \vec{v}_2 \\ \vdots \\ \vec{v}_m \end{pmatrix} = \begin{pmatrix} v_{1,t_1} & v_{1,t_2} & \cdots & v_{1,t_n} \\ v_{2,t_1} & v_{2,t_2} & \cdots & v_{2,t_n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{m,t_1} & v_{m,t_2} & \cdots & v_{m,t_n} \end{pmatrix}$$

Equation 4.1 defines the proportion of a tag t_k after the m-th assignment.

$$p_{m,t_k} = \frac{\sum_{1 \leq i \leq m} v_{i,t_k}}{\sum_{\substack{1 \leq i \leq m \\ 1 \leq j \leq n}} v_{i,t_j}} \tag{4.1}$$

For example, given a set $\mathcal{T} = \{t_1, t_2, t_3, t_4\}$ and three assignments $\{\langle 1,0,0,0 \rangle, \langle 1,0,1,0 \rangle, \langle 1,1,1,0 \rangle\}$, the tag proportions are $p_{3,t_1} = \frac{3}{6}$, $p_{3,t_2} = \frac{1}{6}$, $p_{3,t_3} = \frac{2}{6}$ and $p_{3,t_4} = 0$. After each assignment, the proportions are updated; so, if a new assignment $\langle 0,1,0,1 \rangle$ is performed, the new relative frequencies are $p_{4,t_1} = \frac{3}{8}$, $p_{4,t_2} = \frac{2}{8}$, $p_{4,t_3} = \frac{2}{8}$ and $p_{4,t_4} = \frac{1}{8}$. The values of proportions fluctuate as the number of assignments increases, but asymptotically they tend to stabilize (GOLDER; HUBERMAN, 2006). The set of tag proportions is employed to determine the variation level in the content vocabulary. To illustrate, the analysis of tag proportions in a Delicious trace produced the results shown in Figure 4.1. Next, we present the metric that captures vocabulary variation, and the strategy to adjust the number of allowable concurrent downloads in face of its value.

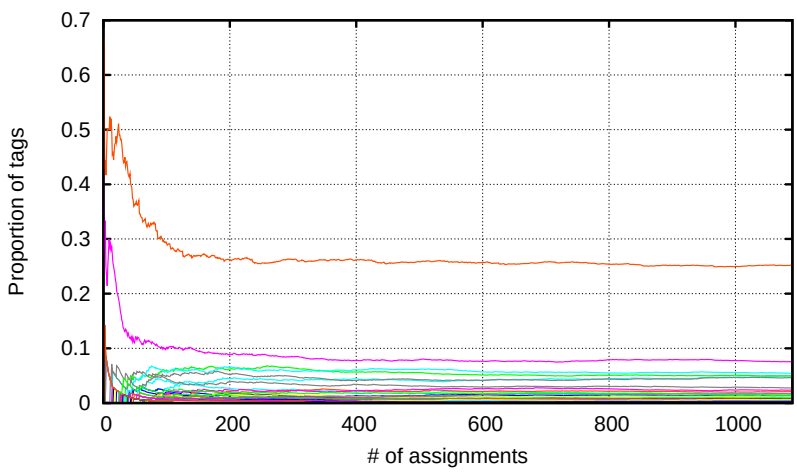


Figure 4.1: Stabilization of tag proportions for a specific Delicious trace

4.1.2 Variation metric

We now present the metric proposed to evaluate different phases in the dynamics of vocabularies. Considering a window with m assignments $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_m$ and a sequence of proportions $\Phi_t = \langle p_{1,t}, p_{2,t}, \ldots, p_{m,t} \rangle$ which reflects these assignments, we define the variation of a tag t in terms of the standard deviation of its proportions ($\sigma_t = \sigma(\Phi_t)$). Several measures of dispersion, such as variance and range, could have been used instead. However, the variance presents a data unit with different scale, equals to the square of the original one, hindering its interpretation. The range, in turn, is not sufficiently robust because it represents an approximation of the dataset variability and is sensitive to boundary values.

To illustrate, the matrix $M_{7,4}$ in the following shows a sequence of 7 assignments (from line 1 to line 7). From the matrix of assignments, we derive a *matrix of proportions* (on the right) whose elements p_{i,t_j} represent the proportion of a tag t_j for the first i assignments, calculated according to Equation 4.1.

$$M_{7,4} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \qquad \left[p_{i,t_j}\right]_{7\times 4} = \begin{pmatrix} {}^1/_2 & {}^1/_2 & 0 & 0 \\ {}^1/_4 & {}^2/_4 & {}^1/_4 & 0 \\ {}^1/_5 & {}^3/_5 & {}^1/_5 & 0 \\ {}^2/_7 & {}^3/_7 & {}^1/_7 & {}^1/_7 \\ {}^2/_9 & {}^3/_9 & {}^2/_9 & {}^2/_9 \\ {}^3/_{13} & {}^4/_{13} & {}^3/_{13} & {}^3/_{13} \\ {}^3/_{14} & {}^5/_{14} & {}^3/_{14} & {}^3/_{14} \end{pmatrix}$$

The vocabulary variation (denoted as Δ_w) is based on a recent subset of tag proportions. In order to capture the current state of the vocabulary after m assignments, a window is used to consider only the w most recent proportions. The Δ_w metric is calculated by summing the variation of each tag (σ_t) within the window, according to Equation 4.2.

$$\Delta_w = \frac{\sqrt{2}}{2} \sum_{t \in \mathcal{T}} \sigma_t \tag{4.2}$$

The constant in the equation is used to normalize the metric. Next, we justify how we have obtained this constant. Since max is a *subadditivity* function, we have that for any non-negative function f, $max(f_1 + f_2 + \cdots + f_k) \leq max(f_1) + max(f_2) + \cdots + max(f_k)$. Thus, in order to find the maximum value for the summation of standard deviations, we must sum the upper bound of each individual term; hence, $max(\sum_{t\in\mathcal{T}} \sigma_t) = \sum_{t\in\mathcal{T}} max(\sigma_t)$. From (JOARDER; LATIF, 2006), we have that $max(\sigma) = \frac{d}{2}\sqrt{\frac{n}{n-1}}$, where d is the sample range and n is the number of elements in the sample ($n \geq 2$). Given that $max(\sigma)$ decreases as n increases, its maximum is reached when $n = 2$; then, $max(\sum_{t\in\mathcal{T}} \sigma_t) = \sum_{t\in\mathcal{T}} \frac{d_t}{2}\sqrt{2} = \frac{\sqrt{2}}{2}\sum_{t\in\mathcal{T}} d_t$, where d_t is the range of the proportions of a tag t. Consider two subsequent assignments ($n = 2$) of two disjoint sets of tags with sizes a and b, respectively. The matrix of proportions is as follows.

$$\begin{pmatrix} \frac{1}{a} & \frac{1}{a} & \overset{\times a}{\cdots} & \frac{1}{a} & 0 & 0 & \overset{\times b}{\cdots} & 0 \\ \\ \frac{1}{a+b} & \frac{1}{a+b} & \cdots & \frac{1}{a+b} & \frac{1}{a+b} & \frac{1}{a+b} & \cdots & \frac{1}{a+b} \end{pmatrix}$$

By summing the absolute difference between proportions of each tag (represented by columns in the matrix), we get $max(\sum_{t\in\mathcal{T}} \sigma_t) = \frac{\sqrt{2}}{2}[(\frac{1}{a} - \frac{1}{a+b})a + (\frac{1}{a+b} - 0)b] = \frac{b}{a+b}\sqrt{2}$.

This equation achieves its maximum if $a = 1$ and $b \to \infty$, which gives a normalization term equals to $\sqrt{2}$.

We illustrate next the use of Equation 4.2 in the matrix of proportions derived from $M_{7,4}$ (suppose $w = 7$, ignoring for the moment the effect of a window):

$$\begin{aligned}
\Delta_7 &= \frac{\sqrt{2}}{2}\left(\sigma(\Phi_{t_1}) + \sigma(\Phi_{t_2}) + \sigma(\Phi_{t_3}) + \sigma(\Phi_{t_4})\right) \\
&= \frac{\sqrt{2}}{2}(0.10436 + 0.10667 + 0.08623 + 0.11196) \\
&= 0.28936
\end{aligned}$$

This means that the aforementioned assignments lead to a variation of approximately 29% in the vocabulary dynamics. This variation is used to adjust the number of allowable concurrent downloads, as explained next.

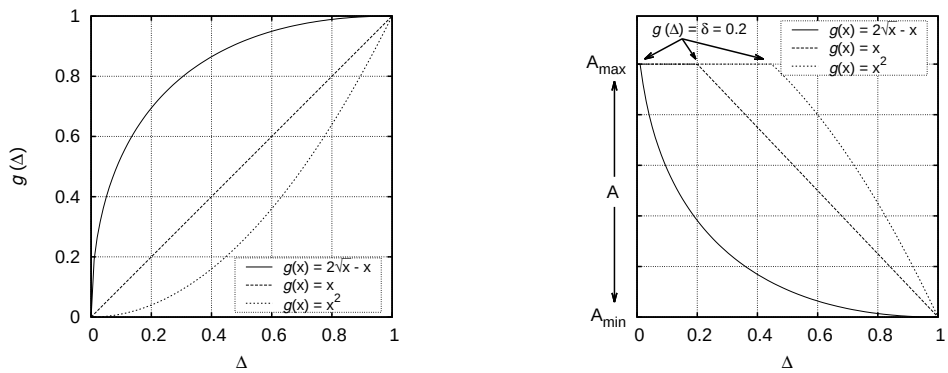
4.1.3 Adjusting the number of allowable concurrent downloads

DÉGRADÉ aims at providing a reliable view of the content vocabulary – i.e., tag proportions have stabilized – reducing the probability that users retrieve contents with poor and insignificant descriptions. Rather than considering only an immediate description of contents, the history of assignments is employed to calculate the vocabulary variation (Δ) and support the strategy to control downloads. In this context, the proposed model reuse the two variables previously introduced by FUNNEL (Subsection 3.1.1): D and A. We recall that D represents the number of downloads currently taking place, whereas A represents the maximum number of concurrent downloads to be allowed. The value of A is defined based on an expected behavior, as described next. Let $\delta \in [0; 1)$ be a threshold parameter, and A_{min} and A_{max} the bounds of A while the strategy is running, we have that:

- If $\Delta \to 1$, then $A \to A_{min}$;
- If $\Delta \to \delta^+$, then $A \to A_{max}$;
- If $\Delta < \delta$, then $A = \infty$.

DÉGRADÉ adjusts the value of A according to Δ. When the variation gradually decreases and converges to δ, A will approach A_{max}. If the variation is below the threshold, then DÉGRADÉ becomes inactive. We introduce a transformation function $g : [0, 1] \mapsto [0, 1]$ that defines how A fluctuates in terms of Δ. For the sake of clarity, Figure 4.2(a) shows three possible functions. The first one produces a more *sensitive* strategy, because small variations (e.g., $0.1 \leq \Delta \leq 0.2$) are mapped to high values (around 0.6). The second function is considered *neutral*, since it does not change Δ. The third function in the figure represents a quadratic function and is considered *resistant* because when the variation metric is close to 0.4, for example, the function returns around 0.2, indicating half of the variation calculated.

The behavior of A is depicted in Figure 4.2(b) for the three aforementioned transformation functions. The threshold δ employed in this example is set to 0.2. When the vocabulary variation is equal to 1, the number of allowable concurrent downloads is A_{min} for all transformation functions shown. As the variation decreases, A approaches A_{max}



(a) A transformation function g maps the variation metric to a new weighted value

(b) Relationship between vocabulary variation and concurrent downloads

Figure 4.2: Impact of different transformation functions on the number of allowable concurrent downloads

with different rates depending on which function is used. When $g(\Delta)$ goes below δ (Δ approximately lower than 0.01, 0.2, and 0.45 for the sensitive, neutral, and resistant curves, respectively), DÉGRADÉ allows every download to proceed. The list of requirements expressed earlier is summarized in Equation 4.3.

$$A = \begin{cases} \infty & \text{if } g(\Delta) < \delta, \\ \frac{(g(\Delta)-\delta)\times A_{min}+(1-g(\Delta))\times A_{max}}{1-\delta} & \text{otherwise.} \end{cases} \tag{4.3}$$

The threshold parameter $\delta \in [0; 1)$ should be adjusted according to restriction levels required by system administrators. If it is set with the lowest value, the condition $g(\Delta) < \delta$ will never be reached and the strategy will always control the number of concurrent downloads. In contrast, as δ approaches 1, less restrictive the strategy becomes, allowing downloads to proceed without control. Thus, A is computed and compared with D; the download is authorized if $D < A$, and delayed otherwise. Therefore, although DÉGRADÉ does not directly change the vocabulary, it reduces the number of downloads being based on low quality (incorrect, uncertain) information. The value of δ along with the transformation function define how conservative the strategy will be. Since we focus the evaluation of our strategy on extreme scenarios, we adopted the sensitive function ($g(x) = 2\sqrt{x} - x$) throughout the analysis. It is important to notice that high variations during the first assignments is an expected behavior, but this does not necessarily mean that a content is mislabeled. Since the proportions among tags vary considerably in early stages of content description, tag assignments may produce high variations in Δ. However, this effect results in a protection against potential attackers when description are more vulnerable to malice, i.e., easier to manipulate. To show the effectiveness of the overall strategy, we carried out an extensive set of simulations, which are discussed next.

4.2 Evaluation

In this section we present and discuss the results obtained with the evaluation of the proposed strategy. First, we describe in detail the traces employed in our evaluation (Subsection 4.2.1) and the scenarios derived from the dataset (Subsection 4.2.2). Then, we present the content distribution fluid-based model used to assess the proposed strategy (Subsection 4.2.3), and a sensitivity analysis performed for a proper setting of the strat-

egy parameters (Subsection 4.2.4). We close the section with a discussion on the results achieved (Subsection 4.2.5).

4.2.1 Dataset details

A fundamental input for the evaluation of our strategy is the set of tag assignments for the content being disseminated. Synthetically generated inputs generally do not capture the true dynamics of users' behavior in face of contents being disseminated. To obtain more accurate and realistic results, the evaluation was based on a set of traces gathered from Delicious, in the context of Tagora Project[1].

The dataset obtained comprises 140,126,586 tag assignments from 532,924 users. These assignments characterize 17,262,480 items (i.e., websites), and involve 2,481,698 distinct tags, summing 1.1 GB of data. A sequence of assignments for a specific item is called a trace. Given its dimension, a methodology was used to select the most relevant set of traces from the dataset, as follows. First, the 2,000 traces with the largest number of assignments were selected. Second, a *clustering algorithm*, called k-medoids, was employed to group these traces in clusters. k-medoids is a partitioning technique that groups a dataset into k clusters. The criterion used to assess the number of clusters is the average "silhouette" of the data, as described by Lletí et al. (LLETÍ et al., 2004). Two significant clusters, from the initial set of 2,000 traces, have been identified according to three properties: (*i*) number of tag assignments, (*ii*) number of distinct tags, and (*iii*) cumulative number of tags. Since our objective was to select representative traces within clusters instead of their properties, the use of the well-known k-means technique did not fit. k-means would select averaged values for the properties, but these values could correspond to no trace from the dataset. Table 4.1 summarizes the clusters and properties of the most representative trace within each cluster (the one whose average dissimilarity to others is minimal, i.e., the *medoid*) within each cluster. For the sake of simplicity, we refer to the clusters as WIDE and NARROW.

Table 4.1: List of clusters

Cluster	# of traces	Properties of the selected traces		
		# of tag assignments	# of distinct tags	Cum. # of tags
WIDE	527	3,001	887	10,195
NARROW	1,473	1,255	335	4,223

4.2.2 Evaluation scenarios

The evaluation of DÉGRADÉ comprised two scenarios, one for each cluster identified according to the methodology presented in the previous subsection. In the remaining of this section, we refer to these scenarios by the cluster name they represent, i.e., WIDE and NARROW. The characteristics, common to both scenarios, are defined as follows.

The process of content dissemination is regarded as a swarm *session*, similar to those in the BitTorrent context. A session starts with I initial seeders, which upload the content indefinitely and remain online during the entire simulation. During the session, C leechers (also referred to as correct peers, i.e., ordinary agents which respect the established protocol) and a proportion of M malicious peers arrive. In order to evaluate DÉGRADÉ against unfavorable conditions, malicious peers arrive before every honest peer.

[1]Available at `http://www.tagora-project.eu/data/#delicious`

Initially, there are 20 seeders ($I = 20$). The proportion of malicious peers varies from 0 to 30% to evaluate the impact of massive attacks. The value of C is set according to each trace, and leechers arrive according to a rate defined by the function $\lambda(t) = \frac{\alpha_0}{1+\beta t}$, whose decay factor β is set to 0.01 and the initial rate α_0 is set in such a way that every peer arrives until $t \approx 4h$ (ANDRADE et al., 2009).

Once a honest peer finishes the download of the content being disseminated, he/she classifies it using one set of tags extracted from the trace that rules the scenario, and remains online as seeders for the entire session. The average number of tags per assignment in both traces is around 3.4. Furthermore, each malicious peer assigns 100 tags following one of the attack strategies depicted in Table 4.2. These attacks are adapted from Koutrika et al. (KOUTRIKA et al., 2008). After the tag assignment, malicious peers leave the system without performing any seeding.

Table 4.2: Attack strategies

Attack	Strategy	Description
RANDOM	Random incorrect tags	Assign a random set of tags not present in vocabulary
TARGET	Fixed incorrect tags	Select a set of tags not present in vocabulary and keep assigning it

The attacks RANDOM and TARGET have the goal of creating a misrepresentation (or incorrect perception) of a given content. By launching such attacks, the attacker attempts to mislead the user in his search for the desired content, and either (i) make him download an undesired content, whose vocabulary erroneously describe another content – the desired one, or (ii) hinder his search for a desired content, by promoting many fake contents whose vocabularies describe the one the user is actually searching for.

It is important to emphasize that the RANDOM and TARGET attacks are the most representative ones in the context of social tagging systems, as evidenced in previous research (KOUTRIKA et al., 2008). This is the reason why we concentrate on these attacks when evaluating the robustness of our strategy.

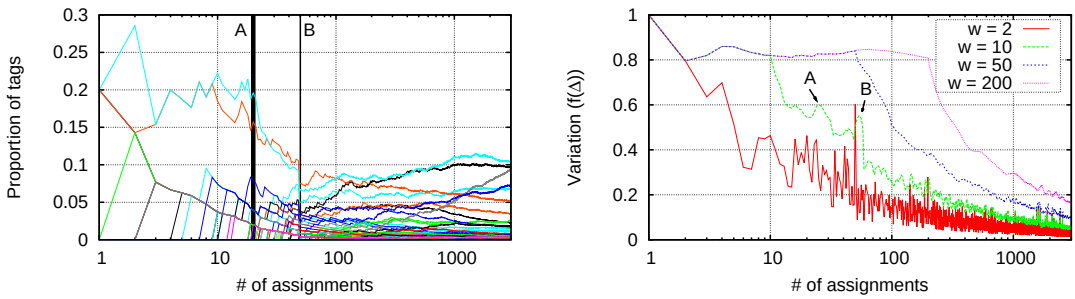
4.2.3 Swarm-based content distribution fluid-based model

Since BitTorrent employs a swarm-based architecture to disseminate content and it is one of the most popular CDS protocols, we have guided our evaluation mimicking a BitTorrent-like model (QIU; SRIKANT, 2004; GUO et al., 2005). It is a fluid model that governs swarms evolution, and allows the analytical estimation of download progresses within time slots. According to the model, the amount of traffic generated in a swarm per time slot can be calculated as $min\{\mu(\eta x + y), cx\}$, where: x is the number of leechers; y is the number of seeders; μ and c are, respectively, the uploading and downloading rates; and η is the file sharing efficiency, which represents the probability of a leecher to upload to another leecher. Qiu and Srikant have demonstrated that for a content divided into P fragments, if each peer connects to other k peers, $\eta \approx 1 - \left(\frac{\log P}{P}\right)^k$ (QIU; SRIKANT, 2004). These equations are used to estimate the number of completed downloads per time slot. For example, considering a content of size F, the i-th peer finishes its download after $i \times F$ is transfered in the swarm.

4.2.4 Sensitivity analysis

The parameters involved in the evaluation were determined through a sensitivity analysis carried out with the selected traces. One of these parameters is the window size (w), which is used by DÉGRADÉ to calculate the vocabulary variation. From a preliminary analysis, we have observed that our strategy becomes more sensitive for small values of w (even in the absence of attacks), and thus may become less effective in capturing the real state of the system. Conversely, higher values of w may obfuscate suspicious behaviors. In order to select a proper value for this parameter, the dynamics of Δ was studied for the various traces available. Figure 4.3 shows a detailed analysis for one trace.

We have observed the proportion of tags to define the window size (w) and the sensibility of DÉGRADÉ. Figure 4.3(a) shows these proportions as a function of the number of assignments. The figure highlights two events in the trace, named *A* and *B*. These events were detected using manual inspection and selected as extreme uncommon behaviors in the trace. The first one ranges from the 20-th to the 25-th assignment, whereas the second one comprises only the 50-th assignment. On event *A*, 31 tags are assigned, out of which 19 have not appeared before in the trace. On event *B*, a single assignment contains 84 tags (whereas 97.5% of assignments contains up to 10 tags each), out of which 53 were new. These events, which we have analyzed and concluded not to be malicious, are considered outliers and must be captured by our variation metric. Figure 4.3(b) shows the impact of different window sizes. For larger values of w (50 and 200), note that Δ remains close to 0.8 despite events *A* and *B*, which are not properly captured. On the other extreme, a very small window ($w = 2$) causes Δ to vary wildly with any event, making it difficult to identify *A* and *B*. We find that an intermediate window size ($w = 10$) is able to capture both events and presents small variations during the subsequent assignments. Thus, $w = 10$ was used in all other simulations.



(a) Stabilization of tag proportions (b) Impact of the window size on variation metric

Figure 4.3: Window size analysis

The other parameters define how restrictive and conservative DÉGRADÉ behaves. Recall that the conservative strategy allows from A_{min} to A_{max} concurrent downloads, depending on how close to δ our variation metric is. An in-depth analysis of δ, A_{min}, and A_{max} was performed to understand their impact on our strategy. With respect to A_{min}, its value should not be high, otherwise there could be massive joins of malicious peers. The value of A_{min}, equal or larger than 1, would ideally reflect the level of popularity of a content. That is, a very popular content might have associated a comparatively higher value of A_{min}, assuming there would be some correct users competing for slots. Since the number of allowed concurrent downloads varies from A_{min} to A_{max} according to δ, if A_{max} is set close to A_{min}, the range of A gets reduced and the strategy becomes more

resistant to the variation metric. Conversely, as we increase A_{max}, the strategy works smoother and gradually adjusts the number of concurrent downloads in terms of δ. We have set A_{min} to a small value ($A_{min} = 10$) to make DÉGRADÉ more robust against colluding attackers present in our scenarios. Moreover, we have made the strategy smoother, with $A_{max} = 3000$, due to high proportions of malicious peers always considered during the evaluations.

Recall from Subsection 4.1.3 that DÉGRADÉ may free all downloads when the variation becomes quite small. The parameter δ determines when DÉGRADÉ does this and becomes inactive. We have investigated two strategies for setting this parameter, namely *relaxed* and *conservative*. The former frees downloads earlier, by accepting a higher δ value (this level of variation can be reached more quickly). This relaxed strategy is adequate for scenarios with few or no malicious users, freeing downloads earlier, but it is more susceptible to attackers manipulation. In contrast, the conservative strategy employs lower values for δ (combined with a large A_{max}) so that the variation needs to be quite small before DÉGRADÉ becomes inactive. In our evaluation, we have chosen the conservative strategy, with $\delta = 0.05$, due to its resilience under attacks.

The last parameter of the model, T, represents the number of time slot peers need to wait before they try again and was set to 5 minutes. The values adopted for the model parameters are summarized in Table 4.3, together with values for the evaluation parameters, mentioned next. The fluid-based model parameters determine the content size (F), the peers' bandwidth (μ and c), and the sharing interest of leechers (η, calculated according to the equation proposed by Qiu and Srikant, with $k = 4$ and the content subdivided in pieces of 16 KB).

Table 4.3: Set of parameters employed in the evaluation

Model parameters	
w	10
δ	0.05
A_{min}	10
A_{max}	3,000
T	5 min

Simulation parameters	
F	512 MB
μ	256 Kbps
c	1,024 Kbps
η	99.99%

We propose three evaluation metrics to capture the effectiveness of our strategy. The first metric, named *similarity*, measures how close the vocabulary is to its final state at the instant users get access to the content being shared. Since the complete trace is available beforehand, we can compare the vocabulary state when users join to the vocabulary resulting from the trace (for simplicity, we denote this final state as "vocabulary definition"). We apply the cosine similarity equation between both vocabularies and get a value between 0 and 1, with values closer to 1 representing stronger similarity. The second metric is the *proportion of "bad" assignments* (i.e., assignments that do not reflect the nature of the content) performed up to the instant when each user gets access to the content (a more stable vocabulary allows the user to make an informed decision). And the last metric is the *downloading time*, which is used to measure the overhead imposed by the strategy and malicious behavior on content downloading time.

These metrics represent an effort on how to measure and quantify the potential quality of experience of users. We are aware that summarizing a subjective aspect into an objective metric requires some sort of simplification, therefore, imprecision. Alternatively, a more robust strategy could be based on survey techniques to capture several aspects of

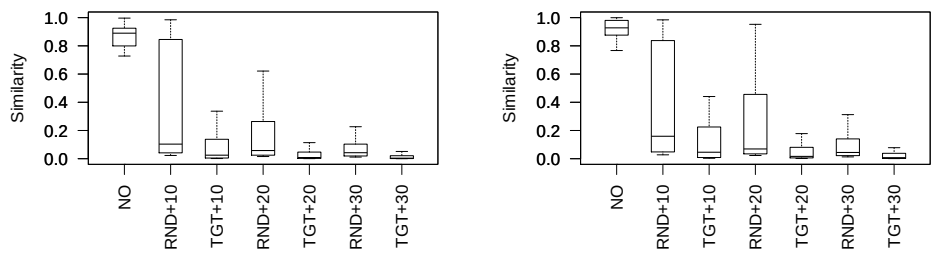
users feedback. Although this would lead to a more precise metric for quality of experience, it would require a laborious work to instantiate it in dynamic systems in the wild. In face of these reasons, we decided to derive objective metrics that are easy to implement and still capture important facets of systems' dynamics.

4.2.5 Results

This subsection aims at answering four research questions, as follows. **Q1.** How harmful content vocabulary attacks can be against systems without a protection strategy? **Q2.** How effective is DÉGRADÉ with and without attacks? **Q3.** What is the overhead imposed by DÉGRADÉ? **Q4.** How is DÉGRADÉ affected by the presence of peers not willing to post testimonies (i.e., assign tags) about downloaded contents?

Impact of attacks on systems running without DÉGRADÉ

In order to understand the effect of the attacks RANDOM and TARGET on content vocabulary (see Subsection 4.2.2), we have carried out simulations using the fluid-based model with our strategy disabled. Figure 4.4 shows the results obtained for the trace clusters WIDE and NARROW (as described in Subsection 4.2.1). Each boxplot, (a) and (b), is composed by seven boxes. Each box represents a distribution of similarity values (on y-axis); hence, the higher the dots concentrate, the better. The first box on the left (labeled NO) in both plots shows the similarity distribution in the absence of attacks and is used as a baseline for comparison. The other boxes show the effect of each attack (RND = RANDOM and TGT = TARGET) for different proportions of malicious users (from 10 to 30%). Observing the plots, first notice that 10% of malicious users performing any of the two attacks are enough to cause significant impact on the similarity metric. This may be easily observed by comparing the median values of the first box with the two subsequent boxes. Note also that the harm caused by attackers increases together with the proportion of attackers. Finally, the plots suggest that the TARGET attack causes slightly more damage than the RANDOM attack, although their median values get closer as the number of malicious users increases.



(a) WIDE (b) NARROW

Figure 4.4: Similarity metric experienced by users when they access contents

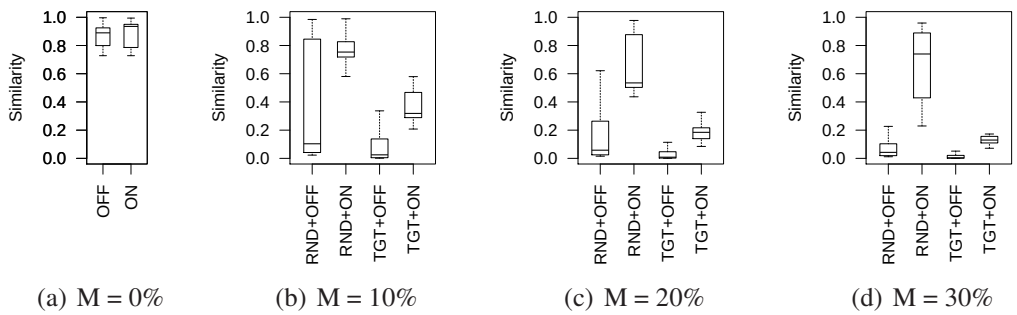
In the absence of DÉGRADÉ, malicious users join the session first and assign "bad" tags according to the respective attack strategy. This behavior negatively impacts the similarity metric, since many users join the session while the current assignments diverge from the vocabulary definition. In scenarios without attack, 50% of users join when the similarity metric is larger than 0.9; in all attack scenarios, the same percentage of users

join when similarity is below 0.2, and frequently, 0.1. These results answer **Q1** by indicating a low level of users' satisfaction when malicious users may freely download (and assign tags to) contents, as honest users access contents with divergent descriptions. Practically, a user may query for a content and receive others as result due to malicious tag assignments. Next, we evaluate the effectiveness of our strategy in hampering attacks and, more importantly, providing users with access to contents whose description is correct and stable.

Effectiveness of DÉGRADÉ *in increasing download quality*

To answer **Q2**, the simulations with attack scenarios (as in the previous subsection) were repeated, but with the strategy on. First, we have observed the behavior of the similarity metric when the strategy is active. Second, we have analyzed the proportion of bad assignments when honest users join the system. DÉGRADÉ will be effective if it is able to achieve a high similarity metric in the absence of attacks, and allows this metric to be minimally affected when the system is under attack. Furthermore, the mechanism activation should enable users to join when there is low proportions of malicious assignments.

Regarding the first part of our observation, Figure 4.5 shows the similarity metric for the WIDE trace obtained without (OFF) and with (ON) DÉGRADÉ. Starting with Figure 4.5(a), notice that the use of DÉGRADÉ without attack provides a small, but noticeable, advantage. More precisely, the similarity median increases from 0.9 to 0.95.



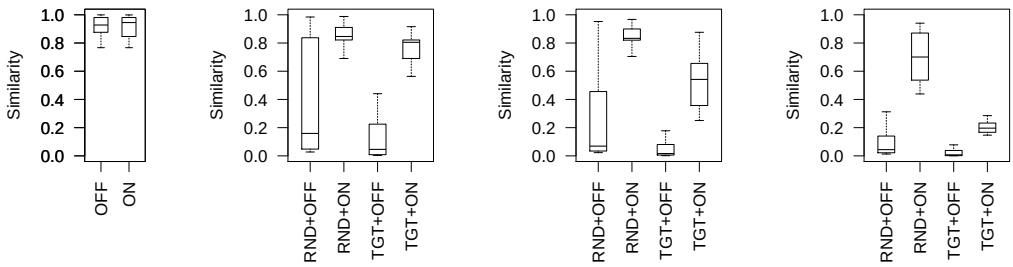
(a) M = 0% (b) M = 10% (c) M = 20% (d) M = 30%

Figure 4.5: Similarity metric experienced by users when DÉGRADÉ is active for different proportions of attackers (WIDE)

As it can be noted in Figures 4.5(b), 4.5(c) and 4.5(d), when attacks strike, DÉGRADÉ is highly efficient in reducing their negative impact. First, considering 10% of malicious users (Figure 4.5(b)), the strategy provides an increase of the similarity median from 0.1 to 0.75 under the RANDOM attack, and from 0.02 to 0.35, under the TARGET attack. When the proportion of malicious users is increased, as shown in Figures 4.5(c) and 4.5(d), the impact of the attacks becomes more severe, but it is clear that in both cases DÉGRADÉ manages to hold the attacks. In particular, we observe that DÉGRADÉ increases the median similarity at least 11 times (from $\sim$0.05 to 0.55) in the case of the RANDOM attack, and at least 15 times (from $\sim$0.01 to 0.15) for the TARGET one.

The aforementioned benefits were seen in the NARROW trace as well (see Figure 4.6). More precisely, we notice that the attacks produce similar effects in both traces when DÉGRADÉ is inactive, but the system using the NARROW trace is slightly less affected. This may be observed by comparing, for example, boxes RND+OFF in Figures 4.5(b)

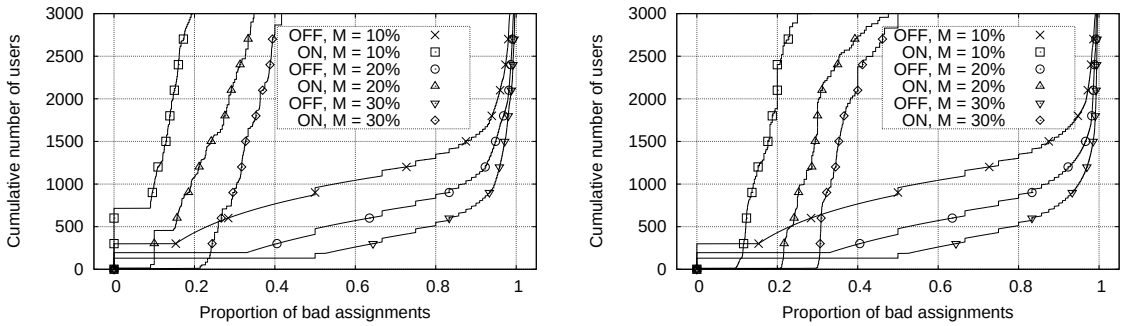
and 4.6(b). Moreover, the advantages of our strategy are highlighted by observing that the NARROW cluster in Table 4.1 contains roughly 73% of all 2,000 studied traces.



(a) M = 0% (b) M = 10% (c) M = 20% (d) M = 30%

Figure 4.6: Similarity metric experienced by users when DÉGRADÉ is active for different proportions of attackers (NARROW)

The second set of simulations performed to answer **Q2** focused on the effectiveness of DÉGRADÉ in preventing malicious assignments, specially in the beginning, when the majority of users arrive. A pair (x, y) in Figure 4.7 means that y users joined the session when there was a proportion of at most x malicious assignments, which means that curves to the left are better than the ones to the right. Each graphic has two curves (OFF and ON) per proportion of malicious users, 10, 20 and 30%. In the absence of DÉGRADÉ (OFF), malicious users are expected to join the session first and assign their "bad" tags. First note that, for each M, the corresponding "OFF" curves in Figures 4.7(a) and 4.7(b) are identical; this is because we measure the proportion of malicious assignments regardless of the attack strategy. More important, the figures show that 10% of attackers are sufficient to produce a huge negative effect: approximately half of the honest users join the session when there have been more than 85% of malicious assignments. As expected, with 30% of malicious users and no control, the attack is more damaging: only 150 honest users (5%) join in the presence of less than 50% of malicious assignments. In addition, half of them experience at least 95% of malicious assignments when they join.

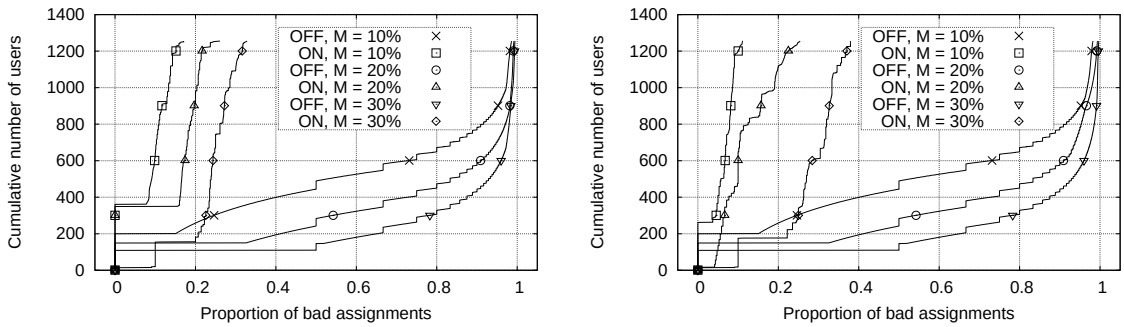


(a) RANDOM (b) TARGET

Figure 4.7: Proportion of bad assignments experienced by users (WIDE)

The effectiveness of DÉGRADÉ is evident for both attacks and traces. When there is 10% of malicious users and the strategy is ON in both attacks, every user joins the session in the presence of at most 25% of malicious assignments (Figure 4.7). Considering the

NARROW trace, this proportion is around 17% of malicious assignments (Figure 4.8). If we increase the proportion of malicious users to 30%, the effectiveness of our strategy is still quite substantial. In the RANDOM attack in both traces, no single user joins in the presence of more than 42% of malicious assignments. The results are similar for the TARGET attack; for example, in the WIDE trace only 500 honest users (∼16%) join in the presence of more than 43% of malicious assignments, but every user joins in the presence of less than 50% of malicious assignments. The protection of DÉGRADÉ against the TARGET attack is noteworthy: every user joins in the presence of at most 38% of malicious assignments.



(a) RANDOM (b) TARGET

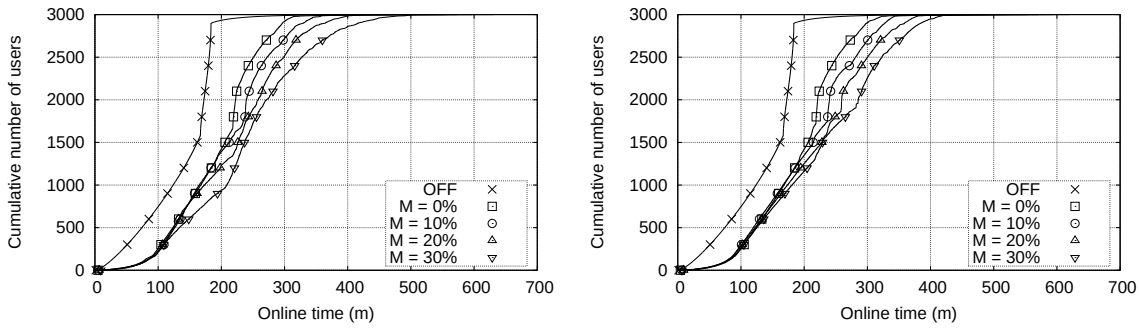
Figure 4.8: Proportion of bad assignments experienced by users (NARROW)

Overhead on users' downloading time

The aforementioned benefits provided by DÉGRADÉ come with a price: a fraction of honest user joins is delayed, until the vocabulary gets more stable. The impact of this overhead, associated with **Q3**, is discussed in this subsection. Figure 4.9 shows the delay imposed by our strategy for different proportions of attackers. A point (x, y) in the plot means that y users take at most x minutes, from their arrival, to complete their downloads. The distance between curves OFF and '$M = 0\%$' represents the overhead of activating DÉGRADÉ in the absence of attacks. The horizontal distance between these curves in Figure 4.9(a) (in Figure 4.9(b) as well) shows that although the overhead can be substantial in some cases, it is less than 50 minutes for most users.

Since DÉGRADÉ delays user joins based on the vocabulary variations, attackers may increase users' downloading time and the overhead on honest users. We observe that the results are similar for both considered attacks. In the worst case for our strategy, attackers can increase downloading times by 30% (by comparing curves '$M = 0\%$' and '$M = 30\%$'). Results for the NARROW trace are omitted due to their similarity to the WIDE one.

In summary, we now have elements to answer **Q3**: the proposed strategy was shown to be effective and efficient in providing a stable view of the vocabulary to users, as well as protecting them from malicious assignments. We also noticed that the impact on users' downloading time is low in the absence of attack, and remains acceptable even when there are 30% of malicious peers. Recall that a system with 30% of malicious users is an extreme case. However, even under these adversarial conditions, DÉGRADÉ has shown to be effective.



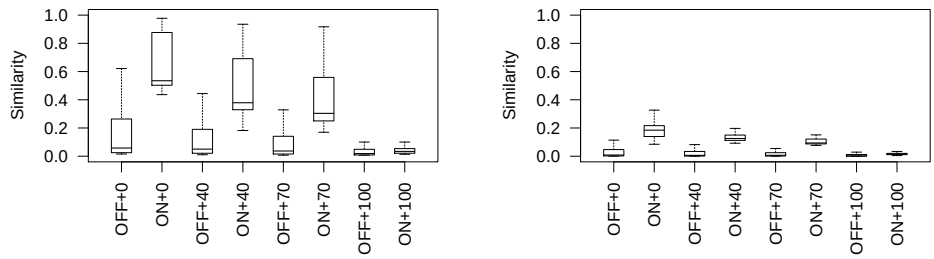
(a) RANDOM (b) TARGET

Figure 4.9: Impact of DÉGRADÉ activation on users' downloading time (WIDE)

Impact of sparse tag assignments on DÉGRADÉ

The previous analysis assumes that all users will assign tags to downloaded content, which may be unrealistic. In practice, only a fraction of users will annotate a content. In this subsection, we evaluate whether sparse tag assignments have a negative impact on the proposed strategy and, if so, quantify it, to answer **Q4**. In all scenarios considered, the proportion of malicious users was fixed at 20% and the proportion of absent users varied between 0 and 100%. Results for both studied traces were similar, hence only the WIDE trace is discussed in this subsection.

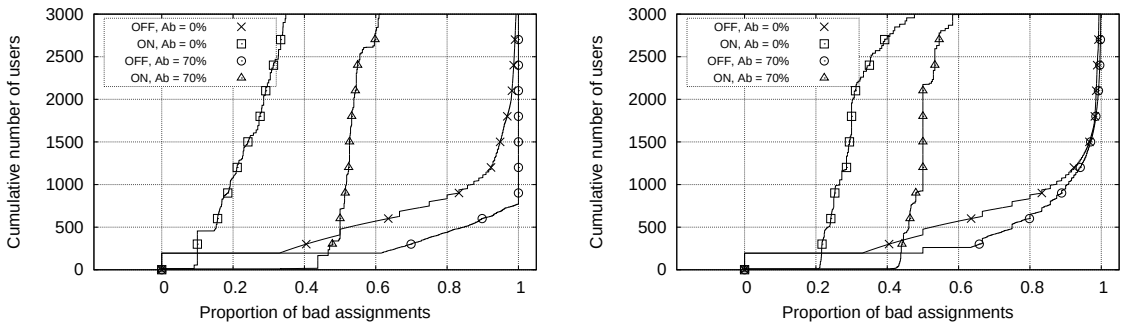
The investigation in this subsection focuses on the three aforementioned metrics. Figure 4.10 shows the impact of absent users on the similarity metric. Each box represents a scenario with a specific proportion of absent users with (ON) or without (OFF) DÉGRADÉ. Results for the RANDOM attack are shown in Figure 4.10(a). As expected, if no honest user assigns tags (boxes OFF+100 and ON+100), users who arrive subsequently experience lower values for the similarity metrics. In scenarios without our mechanism (boxes OFF), users access contents with very poor vocabularies, that is, distinct from their final definition (medians lower than 0.1). In contrast, when DÉGRADÉ is active, results are improved (median greater than 0.3) even when less than one third of users assign tags. Figure 4.10(b) shows that the TARGET attack is quite damaging (with 20% of attackers), leading to low similarity metrics even when there is no absent user. However, when DÉGRADÉ is on, we notice an increase in download quality, since the median increases to 0.2 and 0.1 with 0 and 70% of absent users, respectively.



(a) RANDOM (b) TARGET

Figure 4.10: Similarity metric experienced by users for different proportions of absent users (WIDE and $M = 20\%$)

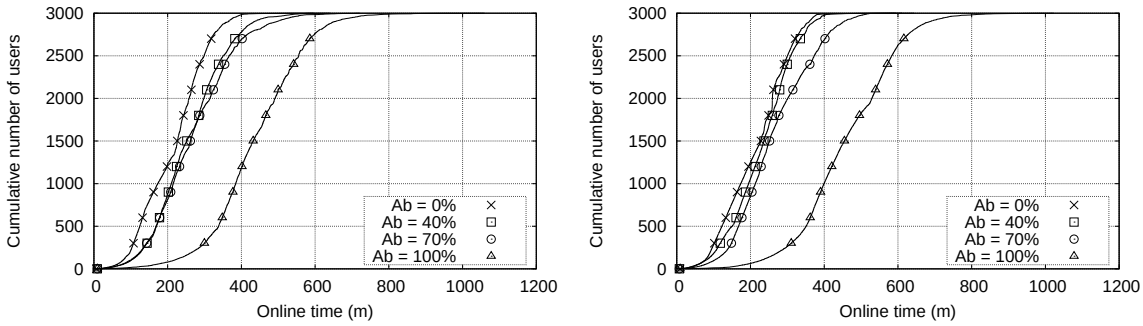
The second evaluated metric is the proportion of bad assignments. Figure 4.11 shows results for both attacks. Scenarios in the absence of honest votes were removed because every user arrives either when nobody voted or when there are only malicious assignments. Figure 4.11(a) presents the evaluation for the RANDOM attack. When DÉGRADÉ is inactive (OFF) and 70% of honest users do not assign tags ($Ab = 70\%$), almost 2,200 users (74%) join the session when there are only malicious assignments. However, the activation of DÉGRADÉ considerably improves the results, enabling almost 2,500 users (85%) to join with 55% or less of bad assignments. Similarly, Figure 4.11(b) presents the evaluation for the TARGET attack. As in the RANDOM attack, results get worse when fewer users assign tags. In the scenario without DÉGRADÉ (OFF) and 70% of absent users ($Ab = 70\%$), 2,000 users join the session with at least 90% of bad assignments. Conversely, when DÉGRADÉ is active, 2,800 users join with at most 55% of bad assignments.



(a) RANDOM (b) TARGET

Figure 4.11: Proportion of bad assignments experienced by users (WIDE and $M = 20\%$)

The third and last metric evaluated to answer **Q4** was download time in scenarios with attack. Since this metric remains unaffected when DÉGRADÉ is off, Figure 4.12 contains only curves pertaining to the use of our strategy. The results are similar for both attacks, RANDOM and TARGET, shown in Figures 4.12(a) and 4.12(b), respectively. We observe that users wait around 50 extra minutes if there is an attack in which 20% of users are malicious and less than one third of honest users assign tags ($Ab = 70\%$). To illustrate the resilience of the proposed strategy, consider one extreme case: with 20% of attackers and no single legitimate tag assignment, the download time is (just) doubled.



(a) RANDOM (b) TARGET

Figure 4.12: Impact of absent users on downloading time (WIDE and $M = 20\%$)

The set of simulations carried out in this subsection aimed at answering how DÉ-

GRADÉ reacts when high proportions of users do not assign tags. The analysis was performed under extreme scenarios in which 20% of users were malicious. Moreover, we also considered the case with 100% of absent users. Results provide evidences to answer **Q4** and show that DÉGRADÉ is effective even when a significant fraction of honest users does not annotate downloaded contents.

4.3 Considerations on the proposed solution

As shown in the previous sections, our strategy provides substantial improvement over existing mechanisms that aim at tackling pollution: while it allows users to express their subjective opinion about contents, it provides a better quality of experience for users that wish to obtain contents in file sharing systems (even in face of malicious behaviors). In this section, we present some considerations regarding the deployment and operation of our strategy in real CDS.

The deployment of DÉGRADÉ (in BitTorrent communities, for example) requires minor changes to the entities that compose it. Depending on the framework design, it is expected that most of the changes be concentrated on the central entities rather than the end-users' entities (this is the case, for example, of centralized and super-peer P2P architectures). Considering its instantiation in a BitTorrent-based file sharing community, the majority of changes required would be concentrated in the *tracker*. These modifications should be done in such a way that the tracker keeps information, for example, about the tag cloud that describes the torrent under its control, and about the number of concurrent downloads of the content pointed by that torrent. Furthermore, the tracker should be modified in order to establish a control upon the number of concurrent downloads, ensure that peers assign tags per torrent once and after download only, and eventually provide incentives for peers that provide feedback after download. While the deployment of these modifications are not expected to violate the basic principles of the BitTorrent protocol, it imposes challenges that should be properly addressed in the future.

With regard to the subjectivity of users' opinion, it is important to emphasize that precisely dealing with such subjectivity is an untraceable problem. Solutions that aim at tackling the content pollution problem in file sharing systems, however, should deal with the subjectivity in users' opinion to be successful. This subjectivity is manifested both when the user publishes a content (and defines the metadata for it), and also when other users express their feedback about the content being shared. In this context, our strategy goes a step further by dealing with the subjectivity using the concept of social tagging systems; this is certainly a significant improvement over existing approaches, which often relied on binary feedback to handle subjectivity. The motivation behind assigning tags may be associated with user's interests. Santos-Neto et al. (SANTOS-NETO et al., 2009, 2013) have shown that individual needs for organizing content provide strong motivation for tagging. Moreover, additional mechanisms may be instantiated to give incentives to users to annotate content (MARLOW et al., 2006; FURNAS et al., 2006; ROINESTAD et al., 2009). In this work, we assume that a tagging system is in place and users cooperate; however, to cover a larger number of possibilities, we also investigated the impact of sparse tag assignments.

Focusing on the conservativeness of our strategy, it might become more conservative with regard to the dissemination of a given content when the metadata precisely reflects the nature of that content. It is important to observe, however, that a robust strategy cannot rely solely on the metadata – which is created by one (or a limited group of) user(s)

when the content is published. For example, over 80% of contents published in KaZaa, and 50% of contents in FastTrack network, have a mismatch between content and metadata (LIANG et al., 2005). Instead, we focus on proposing a variation metric to capture suspicious malicious behavior and use feedback provided by several users – who will consume that content and, therefore, perform a natural manual inspection. We are aware that the variation metric may not capture some facets on vocabulary "health," for example when users naturally change their opinion about something. Moreover, new terms are often created and employed by users to describe contents. Despite these two factors, the metric convergence is just slightly delayed without major impact on the system.

Another important discussion refers to the coexistence of our strategy in file sharing systems that also adopt other mechanisms that limit the number of concurrent downloads taking place. A prospective direction to deal with this aspect is discussed next. First, assume that there are n protocols that estimate their own values for the number of authorized downloads, A_i (with $1 \leq i \leq n$). A solution that combines these protocols should define a function $g : A_1, \ldots, A_n \rightarrow A$, so that A represents an aggregation for the number of allowed concurrent downloads. In order to properly define g, however, it is important knowing the nature of the other protocols. A prospective direction to define g would be using the average of $A_1, \ldots, A_n$. Different weights might also be used for each value of A_i. It is important to emphasize, however, that a real implementation needs to consider the nature of the file sharing system as well.

4.4 Summary

Content pollution in CDS is a topic that has received considerably attention from the research community. In general, proposed strategies are based on users' feedback for reducing the reputation of corrupted, mislabeled contents, as well as those containing virus or other malware. However, these strategies have neglected the subjectivity of users' opinions regarding contents. Disregarding such subjectivity, users may download undesired content, eventually leading to users' dissatisfaction with the community and even waste of resources (such as time and network bandwidth). To address this limitation, in this chapter we presented DÉGRADÉ, a tag-based strategy to hinder the dissemination of undesirable contents in file sharing systems.

The simulations carried out showed the effectiveness of using tags as a mechanism to control the dissemination of contents, when there was insufficient/imprecise information describing it. By tracking the variation of the proportion of tags that formed the vocabulary describing that content, fewer peers were authorized to proceed their downloads – since there was a higher probability that the content would not actually correspond to what users really expected.

Another positive aspect of DÉGRADÉ – observed in consequence of the delay imposed to users' downloads – is that it enabled a more precise construction of the content vocabulary. As download requests were gradually authorized, attacks *en masse* were thwarted by DÉGRADÉ; further, honest peers also arrived in the system and got a chance to download the content.

The experience obtained during the design of FUNNEL and DÉGRADÉ helped us to find many other domains that may take advantage of conservative strategies. Instead of controlling user's joins, these strategies may also be applied to delay malicious actions in general. To explore these possibilities and increase the overall strategy applicability, we derived a generic model from our previous solutions. The next chapter presents this

model and discusses some examples where this novel solution may improve user's quality of experience.

5 GENERALIZATION OF THE MODEL

This chapter presents a generic model to mitigate massive attacks in CDS. A clear benefit of designing a novel generic model is that the resulting solution tackles not only content pollution problems, but also other denial-of-service (DoS) attacks. The previous models, FUNNEL and DÉGRADÉ, are generalized and the artificial delay imposed to users is determined by a set of mathematical functions. Users (peers) are henceforth referred as *consumers*, and CDS and *service* are used interchangeably throughout this chapter. Thus, the overall system comprises three main entities: service, contents, and consumers. The *service* is responsible for storing and indexing all available contents. *Contents*, in turn, are objects of interest that consumers wish to download. *Consumers* are users of the system. They access the service to search for and download contents.

Similarly to scenarios investigated in previous chapters, there is a CDS in which consumers may insert and/or describe contents according to their own perceptions. We recall that this degree of freedom enables attackers to publish fake contents and provide imprecise metadata about existing ones. Although isolated malicious actions are commonly tolerated by existing systems, massive interference of attackers potentially decreases the quality of experience (QoE) of consumers and may turn the system into a useless content distribution platform. The solution presented in this chapter introduces the concept of *delaying functions* to protect consumers against malicious behaviors and, hence, increase their overall QoE. The usage of delaying functions is employed to guide the number of concurrent downloads allowed in the system (represented by A in FUNNEL and DÉGRADÉ approaches). As users join and assign votes (or tags), more or less downloads are authorized. Thus, to generalize our previous models, the delaying function would consider these votes to impose short or long delays to users. Delaying users decreases the chance that large number of attackers takeover the system and affect users' QoE. In the context of this work, QoE can be defined in terms of the amount of attackers who join the system before each consumer. This represents the potential damage caused by malicious behaviors, since subsequent arriving consumers will experience a service potentially tampered by attackers, e.g., unbalanced votes issued, imprecise descriptions about contents, and network topology manipulation.

The proposed strategy employs a delaying mechanism so as to allow a large proportion of honest users to arrive in the service and compete with malicious ones (that potentially arrive earlier) in order to get access to the service. Given that (*a*) there is a larger proportion of honest users than malicious ones waiting for a chance to access the system and (*b*) the strategy randomly selects which users will join at a given instant, it is possible to protect the system from massive damage that would be caused by attackers in the early stages of the content distribution process. Legal aspects of shared content are out of the scope of this thesis and, hence, the conservative strategy aims at providing users

what they expect to find (ideally free of malicious interferences). To prove concept and technical feasibility of the proposed strategy, a fluid-based model is formulated (capturing users' arrival, delaying functions etc) and an extensive set of simulations is carried out to understand how such strategy performs considering both realistic and worst-case scenarios. A discrete-event simulator is also instantiated to overcome the limitations of the analytic model and investigate the impact of the proposed strategy on individual users.

Organization. The remainder of this chapter is organized as follows. Section 5.1 formalizes massive attacks as a superset of content pollution attacks. Section 5.2 introduces the conservative strategy to delay users and the QoE metric used to evaluate the solution. Section 5.3 presents a detailed evaluation in multiple use cases. Section 5.4 draws some considerations on the proposed solution. Finally, Section 5.5 closes the chapter with a summary.

5.1 Problem formalization

In scenarios considered in this chapter, there is a service that hosts contents and a set of users (*consumers* and *attackers*) who want to join the system. Consumers try to access content and download it, whereas attackers aim at promoting poor QoE to consumers, e.g., posting wrong testimonials about content, publishing content with imprecise descriptions, flagging authentic content as inappropriate etc. In order to represent the behavior of users, a set of mathematical functions determines the amount of consumers and attackers arriving in the system per time unit. For the sake of generalization, a fluid-based model is introduced so that continuous functions are applied to describe accesses to contents.

At any given instant t, let functions $\lambda_c(t)$ and $\lambda_a(t)$ represent the arrival rate of consumers and attackers, respectively. Given these arrival rate functions, the definite integral from m to n, for example, represents the total number of users who arrived during the interval $[m, n]$. Therefore, Equations 5.1 determine the total number of consumers and attackers in the system until instant t_f, i.e., $m = 0$ and $n = t_f$.

$$c(t_f) = \int_0^{t_f} \lambda_c(t)\,\mathrm{d}t \qquad a(t_f) = \int_0^{t_f} \lambda_a(t)\,\mathrm{d}t \tag{5.1}$$

To measure the QoE in the system, it is necessary to introduce a function that represents the proportion of attackers in the system at any given instant t. Equation 5.2 combines both c and a to formalize this proportion, called p.

$$p(t) = \begin{cases} \frac{a(t)}{c(t)+a(t)} & \text{if } c(t) + a(t) > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{5.2}$$

To calculate the QoE at instant t_f, defined in terms of the average proportion of attackers when consumers join the system, it is necessary to accumulate p for every arriving consumer λ_c and divide it by the total number of consumers $c(t_f)$. This metric, henceforth named $\mathcal{Q}$, measures the overall QoE in the system and is calculated as shown in Equation 5.3.

$$\mathcal{Q} = 1 - \frac{1}{c(t_f)} \int_0^{t_f} \lambda_c(t) p(t)\,\mathrm{d}t \tag{5.3}$$

Given the functions that reproduce user arrivals and the QoE metric, it is possible to envisage a conservative strategy to "reshape" the arrivals and mitigate massive malicious

joins in the system. The next section introduces the concept of delaying functions to protect users and improve their QoE.

5.2 Conservative strategy

Before introducing the generic model to tackle general massive attacks, it is important to recall important facets of DÉGRADÉ and FUNNEL models. In FUNNEL model, binary votes are employed to calculate a content reputation. This reputation is employed to determine the number of concurrent downloads to be authorized. In the absence of positive and negative votes, DÉGRADÉ model considers the stability in content vocabulary to derive a variation metric. Similar to the content reputation used by FUNNEL, this metric is employed to determine the number of downloads to be authorized. In summary, both models employ user's feedback to adjust the number of concurrent downloads.

Essential elements present in FUNNEL and DÉGRADÉ are summarized in Table 5.1. The *vocabulary* describes how users express their opinion about contents (e.g., votes or tags). The objective is to provide enough information to calculate a numeric value that represents "content health". This value, called *metric*, is employed to determine the number of allowed concurrent downloads. As the metric approaches the *threshold*, the conservative strategy assumes content as properly classified and stops controlling it.

Table 5.1: Comparison between our conservative strategies

	FUNNEL	DÉGRADÉ
Vocabulary	binary	free
Metric	reputation (R)	variation (Δ)
Threshold	r	δ

In order to generalize our conservative strategy, the metric and the threshold are replaced by a delaying function which determines the number of users allowed to join the system at a given instant. This function forces users to wait and increases the probability that consumers join before attackers, hence, protecting the system against malicious manipulations in the early stages of the content distribution process. Two terms are employed with distinct meanings: while the term *arrive* refers to the act of arriving in the system to access some content, *join* is used when the access is actually granted.

Since the control strategy randomly selects users to access the system, it is known that the probability of a consumer or an attacker to be granted access to the system is directly related to their respective proportions waiting to join. Given the number of consumers and attackers waiting to join at instant t, represented by $w_c(t)$ and $w_a(t)$ respectively, the probability that a consumer or an attacker joins (C and A) is defined by the following Equations 5.4.

$$\begin{cases} P[C] = \frac{w_c(t)}{w_c(t)+w_a(t)}, P[A] = 1 - P[C] & \text{if } w_c(t) + w_a(t) > 0, \\ P[C] = P[A] = 0 & \text{otherwise.} \end{cases} \tag{5.4}$$

Let $f(t)$ be a delaying function that represents the rate at which users are allowed to join the system at instant t. Equation 5.5 employs $P[C]$ to determine the number of consumers actually joining the system, $\overline{\lambda_c}$. Similarly, Equation 5.6 employs $P[A]$ to determine the number of attackers joining, $\overline{\lambda_a}$.

$$\overline{\lambda_c}(t) = \begin{cases} P[C] \times f(t) & \text{if } w_c(t) + w_a(t) > f(t), \\ w_c(t) & \text{otherwise.} \end{cases} \tag{5.5}$$

$$\overline{\lambda_a}(t) = \begin{cases} P[A] \times f(t) & \text{if } w_c(t) + w_a(t) > f(t), \\ w_a(t) & \text{otherwise.} \end{cases} \tag{5.6}$$

The overall system design is illustrated in Figure 5.1 through a fluid flow model. For the sake of representation, the figure illustrates a snapshot of the system. The two uppermost recipients represent the sources of consumers (on the left) and attackers (on the right). The drops fall into the funnel according to known arrival rates. The liquid accumulates in the funnel and leaves according to the f function. The final destination is the recipient in the bottom, where consumers and attackers "stay" after they have joined.

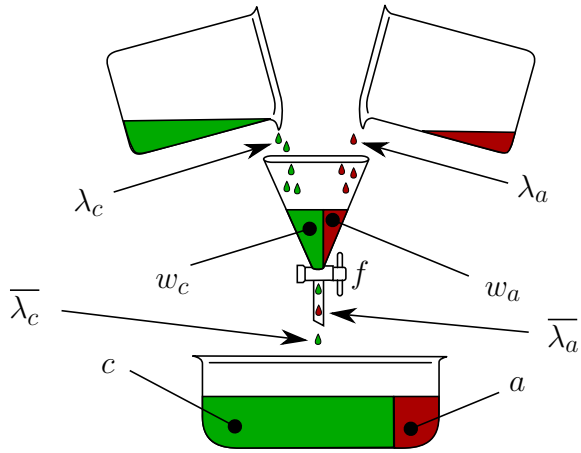


Figure 5.1: Overall system design

In order to measure the overhead imposed to the N first consumers due to the delaying strategy, a metric called *average waiting time* ($\mathcal{W}$) is introduced. This metric is calculated by subtracting the times each consumer arrives from the times they join in the system and dividing this difference by N. Let tf and $\overline{tf}$ be the times when the N-th consumer arrives and joins the system, respectively. The definite integral of arrival and joining times multiplied by the corresponding functions, λ_c and $\overline{\lambda_c}$, is employed to calculate $\mathcal{W}$, as shown in Equation 5.7.

$$\mathcal{W} = \frac{1}{N}\left(\int_0^{\overline{tf}} t\overline{\lambda_c}(t)\,\mathrm{d}t - \int_0^{tf} t\lambda_c(t)\,\mathrm{d}t\right) \tag{5.7}$$

Distinct delaying functions produce different QoE ($\mathcal{Q}$) and average waiting time ($\mathcal{W}$). For example, a delaying function can be more restrictive by allowing only a few users to join in the beginning. In this case, users who arrive early experience long waiting times, but are more protected against attackers. Conversely, if the delaying function is more permissive in the beginning, users (including potential attackers) are allowed to join earlier and waiting times will be shorter. The trade-off between $\mathcal{Q}$ and $\mathcal{W}$ is investigated and discussed throughout the evaluation section.

5.3 Evaluation

This section instantiates the aforementioned strategy in order to evaluate its use in multiple scenarios and under adversarial conditions. First, Subsection 5.3.1 presents the set of parameters and the evaluation scenarios. Second, Subsection 5.3.2 investigates the worst and best cases to be used as baselines in the evaluation. Third, Subsection 5.3.3 discusses analytical results for two scenarios and presents an in-depth investigation about waiting times.

5.3.1 Evaluation scenarios

In order to investigate the impact of different delaying functions on the $\mathcal{Q}$ and $\mathcal{W}$ metrics, two baselines were designed: (*i*) without delaying functions, called WORSTQ, and (*ii*) with a delaying function that forces users to wait until all of them have arrived, called BESTQ. The first case depicts the minimum possible overhead, i.e., consumers do not need to wait, but this produces the worst value for the $\mathcal{Q}$ metric. In contrast, the second case leads to the best value for $\mathcal{Q}$.

Two other cases were designed to evaluate the strategy. The first one illustrates a scenario in which consumers arrive uniformly within 1 hour, called uniform (UN) for short. The second case, in turn, illustrates a flash crowd scenario (FC), in which consumers arrive within 24 hours, but 80% arrive in the first 5 hours. In both scenarios, *C* consumers and *A* attackers arrive in the system. The arrival and delaying functions are defined in terms of users per second.

Two delaying functions were used to evaluate each scenario: CONSTANT and LINEAR. The CONSTANT function enforces a constant joining rate, whereas the LINEAR one enforces a linear rate, both adjusted by an α coefficient. In this evaluation, *C* is set to 3,000 and *A* varies so that attackers represent from 0 to 30% of all users in the system. Moreover, attackers always arrive *en masse* before every consumer. The parameters used in this evaluation section are summarized in Table 5.2.

Table 5.2: Set of parameters employed in the evaluation

C	3,000	
A	$0 \ldots 30\%$	
$\lambda_c(t)$	3,000/3,600 (UN, 1 hour)	$33.097/(1+0.1t)$ (FC, 24 hours)
$f(t)$	α (CONSTANT)	αt (LINEAR)

5.3.2 Baseline

As previously mentioned, two boundary cases were designed to support the evaluation of different delaying functions. For the first case, called WORSTQ, Equation 5.7 was solved for $\mathcal{W} = 0$, which yields $\overline{\lambda_c} = \lambda_c$. Since attackers arrive first, every consumer experiences the maximum number of attackers in the system. Conversely, for the second case, called BESTQ, every consumer and attacker is waiting before they are allowed to join the system. Therefore, the value of p in Equation 5.3 remains constant, regardless of $\overline{\lambda_c}$, and produces the best value of $\mathcal{Q}$ for a specific number of attackers. In order to perform a fair comparison across multiple scenarios, a normalized $\mathcal{Q}$ is introduced ($\mathcal{Q}_{norm}$) as a scaled value of $\mathcal{Q}$ between WORSTQ and BESTQ. Thus, for different proportions of

attackers, the value of $\mathcal{Q}_{norm}$ decreases to 0 as $\mathcal{Q}$ approaches WORSTQ and increases to 1 as it approaches BESTQ. The values of WORSTQ and BESTQ employed to calculate $\mathcal{Q}_{norm}$ are summarized in Table 5.3.

Table 5.3: Boundary values for the baselines

A	WORSTQ	BESTQ
10%	.74	.90
30%	.48	.70

5.3.3 Results

Two scenarios (UN and FC) are investigated using different delaying functions: CONSTANT and LINEAR. The values of α coefficient for both functions vary to produce different results for $\mathcal{Q}_{norm}$ and $\mathcal{W}$. Furthermore, an in-depth investigation using discrete-event simulation (DES) is carried out to measure the overhead on individual consumers.

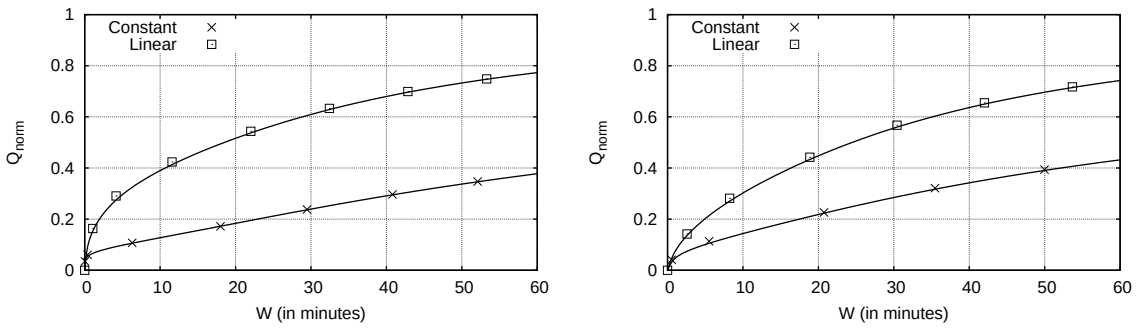
Uniform scenario

In the uniform scenario, 3,000 consumers arrive uniformly distributed during 1 hour. However, before they arrive, a proportion A of attackers arrive in the system. In the context of this evaluation, two values of A are used, 10 and 30%. Figure 5.2 shows the relationship between $\mathcal{W}$ and $\mathcal{Q}_{norm}$. As expected, consumers experience a better $\mathcal{Q}_{norm}$ when their average waiting time increases.

In both cases, with 10 and 30% of attackers (Figures 5.2(a) and 5.2(b), respectively), the LINEAR function produces better results when compared to the CONSTANT one, i.e., for any given $\mathcal{W}$, the LINEAR curve is above the CONSTANT. Moreover, Figure 5.2(a) shows that in the presence of 10% of attackers, when the average waiting time is 30 minutes, the LINEAR delaying function results in $\mathcal{Q}_{norm}$ equals to 0.6. Even when there is 30% of attackers, as shown in Figure 5.2(b), $\mathcal{Q}_{norm}$ is greater than 0.5.

It is possible to notice that there is a clear relationship between $\mathcal{W}$ and QoE. Users from different kinds of systems may tolerate different waiting times. It is unlikely to happen that users wait one hour to watch a 30 seconds video, for example. In this case, users may prefer to try different versions of the video until they find the expected one, unless this comprises a cumbersome task that takes approximately one hour. Considering a case in which users need to download a large file before using it, they may prefer to wait longer in order to avoid undesirable content, hence saving resources.

To understand the QoE each consumer faces individually, it is necessary to observe the proportion of attackers in the system when each consumer joins. To focus on an extreme case, Figure 5.3 illustrates the system behavior in the presence of 30% of attackers. In a first turn, three curves are plotted in Figure 5.3(a): one represents the curve without a delaying function, and the other two represent the curves with the CONSTANT and LINEAR functions. A point (x, y) in the plot means that y consumers join when there is a proportion of x, or less, attackers in the system. In a second turn, these cases are detailed in Figures 5.3(b), 5.3(c), and 5.3(d). The value of the α coefficient was set so that the average waiting time of consumers was 30 minutes. The curve 'OFF' in Figure 5.3(a) shows that 2,000 consumers join when there is already more than 40% of attackers in the system. In contrast, less than 1,200 consumers experience more than 40% of attackers when the CONSTANT strategy is in place. This number of consumers is reduced to less than 800 when the strategy is LINEAR.



(a) $A = 10\%$ (b) $A = 30\%$

Figure 5.2: Relationship between $\mathcal{W}$ and QoE (UN scenario)

Figures 5.3(b), 5.3(c), and 5.3(d) plot the cumulative number of consumers and attackers who joined the system. The axes x and y represent the time in minutes and the number of users, respectively. In the absence of a delaying function (Figure 5.3(b)), all the attackers are in the system from the beginning and consumers arrive following a uniform distribution. The scenario using the CONSTANT function (Figure 5.3(c)) illustrates these attackers being delayed and consumers slowly joining. Similarly, using the LINEAR function (Figure 5.3(d)), the number of consumers is greater than the number of attackers most of the time. It is important to emphasize that each plot contains an extreme proportion of 30% of attackers.

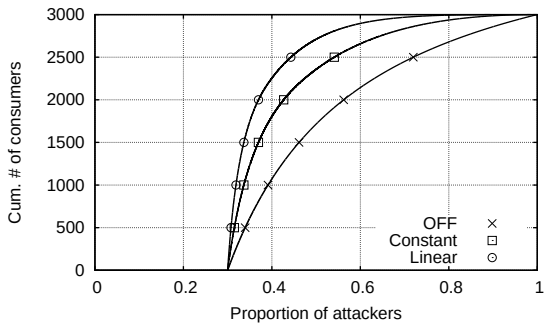
Summary. The use of delaying functions produced an improvement on $\mathcal{Q}$. The LINEAR function led to lower $\mathcal{W}$ and higher $\mathcal{Q}$ in comparison to the CONSTANT one. Although consumers arrive during 1 hour in the studied uniform scenario, results are valid also for any other scenarios in which arrivals are uniform.

Flash crowd scenario

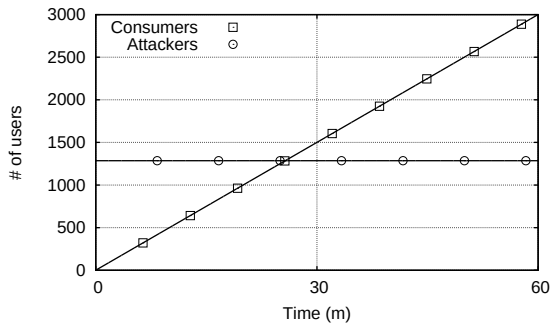
In the flash crowd scenario, almost 2,500 consumers arrive in the first 5 hours. In total, 3,000 consumers arrive within 24 hours. As in the previous scenario (UN), the attackers arrive first in the system. Figure 5.4 shows the relationship between $\mathcal{W}$ and $\mathcal{Q}_{norm}$ for 10 and 30% of attackers. The first observation is that although the difference between $\mathcal{Q}_{norm}$ is not greater than 0.1 for a given $\mathcal{W}$ in CONSTANT and LINEAR functions, the latter again performs better than the former.

Figure 5.4(a) shows that in the presence of 10% of attackers, the CONSTANT function increases the $\mathcal{Q}_{norm}$ metric to 0.8 when the average waiting time is 5 hours. This value is close to 0.9 for the LINEAR function. A similar effect is observed in Figure 5.4(b), in which 30% of attackers arrive in the system. However, the increment in the number of attackers decreases $\mathcal{Q}_{norm}$ from 0.9 to 0.8 for the LINEAR function ($\mathcal{W} = 5$), but it is still a significant improvement.

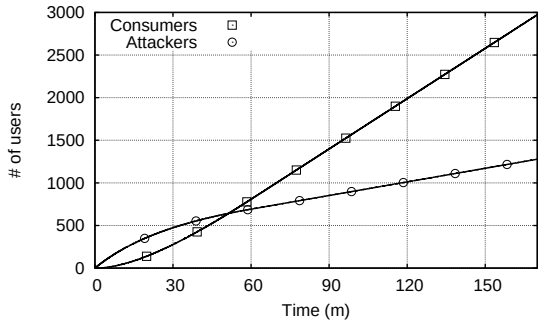
To investigate the system behavior in more detail, Figure 5.5 focuses on results for 30% of attackers. Figure 5.5(a) shows the proportion of attackers in the system when consumers join. In this evaluation scenario, α was set so that the average waiting time of consumers was 12 hours. The magnitude of waiting times is system-dependent and its feasibility is discussed later in this section. The curve 'OFF' is equal to that in Figure 5.3(a), since all the attackers are already in the system and consumers' arrival rate does not matter in this case. However, both CONSTANT and LINEAR delaying functions significantly decrease the proportion of attackers when consumers join, i.e., around 80%



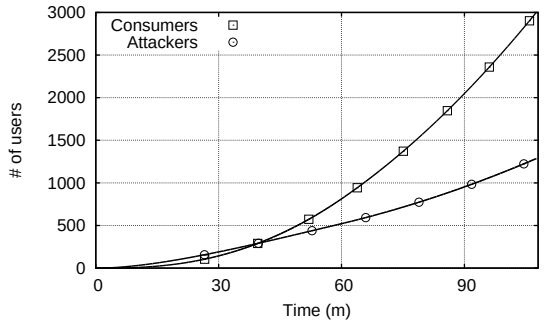
(a) Proportion of attackers in the system when consumers join



(b) Number of users (OFF)



(c) Number of users (CONSTANT)



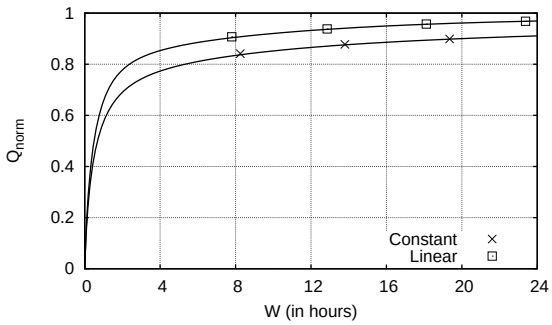
(d) Number of users (LINEAR)

Figure 5.3: Dynamics of users in the system in the presence of 30% of attackers (UN scenario)

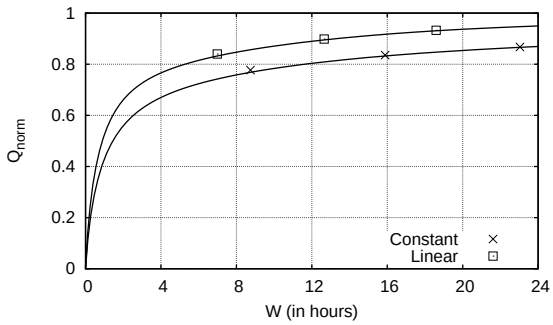
of consumers join when there is less than 35% of attackers.

In the absence of a delaying function (Figure 5.5(b)), all the attackers are in the system from the beginning and the number of consumers increases quickly in the first minutes (flash crowd). Figures 5.5(c) and 5.5(d) show that CONSTANT and LINEAR functions, respectively, delay users so that consumers represent majority of users all the time in the system. Moreover, the number of attackers is severely reduced in the early phases, i.e., instead of 1,285 attackers (30%) since the beginning, there are only 500 attackers after 16 hours using CONSTANT or LINEAR function.

Summary. In flash crowd scenarios, the number of consumers rapidly increases and even low values of $\mathcal{W}$ are enough to significantly improve $\mathcal{Q}$. Although both CONSTANT

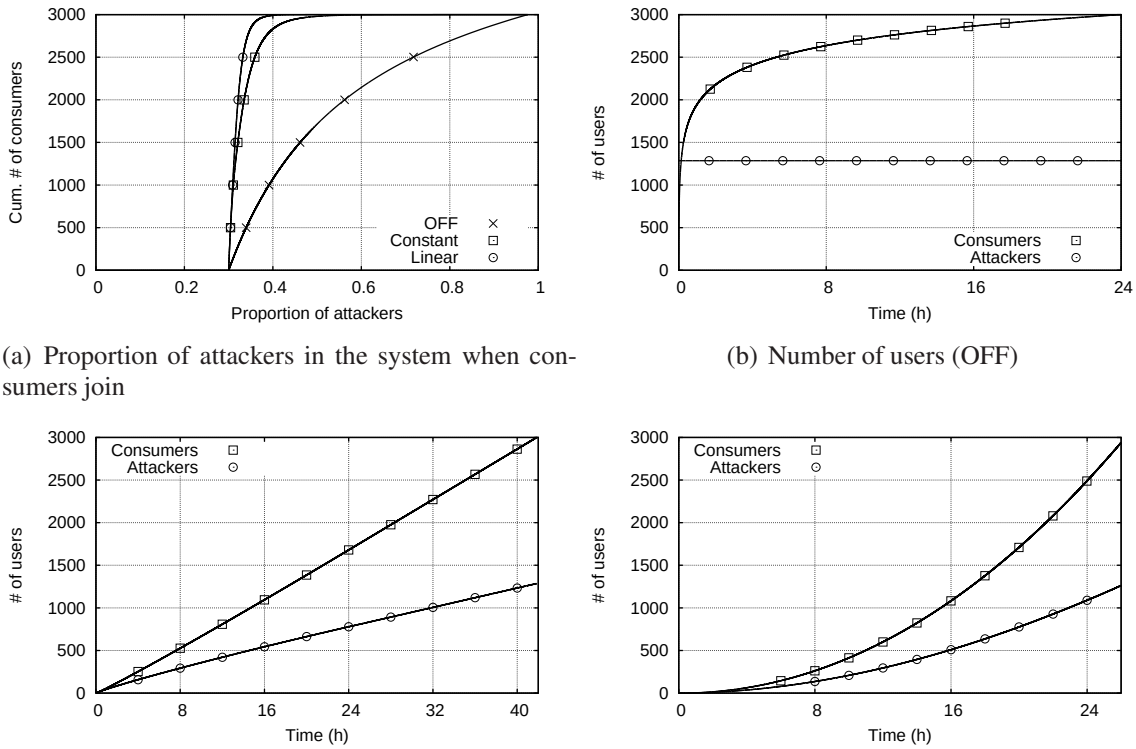


(a) $A = 10\%$



(b) $A = 30\%$

Figure 5.4: Relationship between $\mathcal{W}$ and QoE (FC scenario)



(a) Proportion of attackers in the system when consumers join

(b) Number of users (OFF)

(c) Number of users (CONSTANT)

(d) Number of users (LINEAR)

Figure 5.5: Dynamics of users in the system in the presence of 30% of attackers (FC scenario)

and LINEAR functions achieve similar results for $\mathcal{Q}$, the former provides higher values for QoE with lower waiting times.

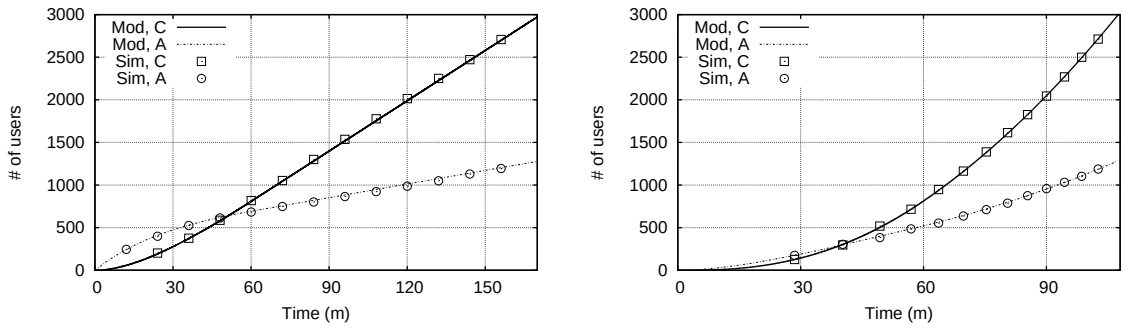
Waiting time analysis

The results discussed earlier were obtained using the analytic model presented in Sections 5.1 and 5.2. However, the model provides an average ($\mathcal{W}$) and thus cannot capture details about the waiting time of individual users. To circumvent this limitation, a discrete-event simulation (DES) was carried out to reproduce previous results. Four scenarios were simulated: UN+CONSTANT, UN+LINEAR, FC+CONSTANT, and FC+LINEAR, each one containing the extreme proportion of 30% of attackers.

In the simulator, consumers and attackers arrive exactly as explained in Subsection 5.3.1, i.e., consumers arrive uniformly (UN) or in flash crowd (FC), and attackers always arrive first. The delaying functions determine the number of users allowed to join. Users are *randomly* selected by the strategy from the set of waiting users. The waiting time for each user is computed by subtracting the joining and arrival times.

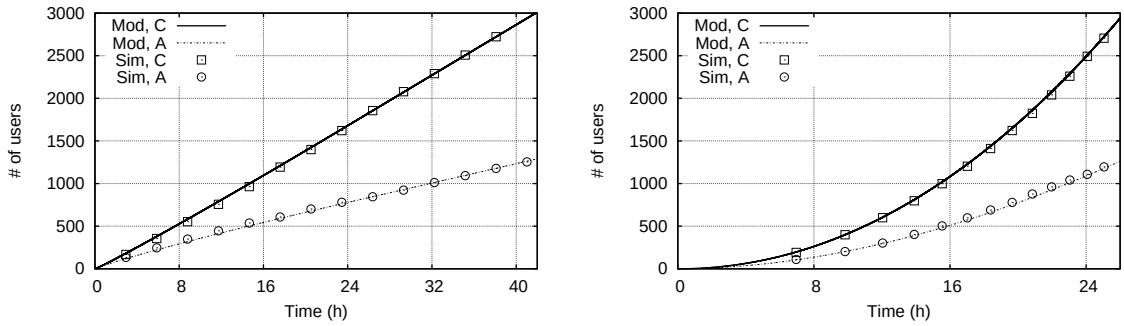
Figures 5.6 and 5.7 show the scenarios UN and FC, respectively. Lines in the plots represent previous results obtained using the analytic model (Mod) and points represent simulation results (Sim). The characters C and A stand for consumers and attackers, respectively. Results show that curves (model) and respective points (simulation) match for each evaluated case, which indicates that the simulator behaves exactly as expressed in the analytic model. Simulation results are discussed to provide the reader with details of what individual consumers experience when trying to access the service.

Given that users need to wait for a chance to join the system, it is of paramount im-



(a) CONSTANT delaying function (b) LINEAR delaying function

Figure 5.6: Validation of the simulator when the conservative strategy is running in the presence of 30% of attackers (UN scenario)



(a) CONSTANT delaying function (b) LINEAR delaying function

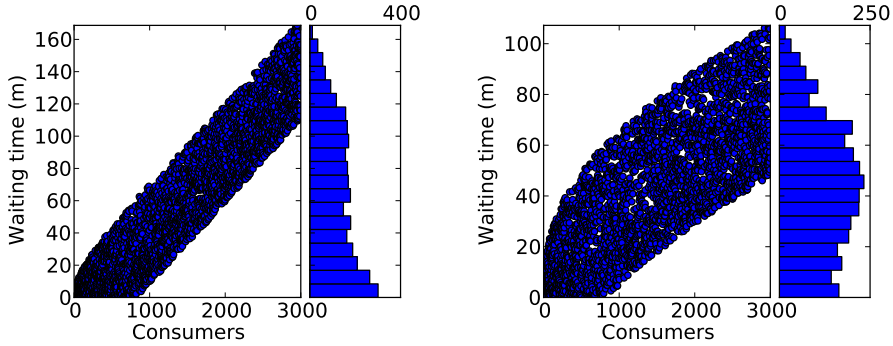
Figure 5.7: Validation of the simulator when the conservative strategy is running in the presence of 30% of attackers (FC scenario)

portance to observe not only the average waiting time, but also the impact on individual consumers. To illustrate that, Figures 5.8 and 5.9 plot consumers' waiting times paired with histograms for UN and FC scenarios, respectively. Each point (x, y) represents the waiting time y of a consumer x. Consumers are sorted on x-axis according to their arrival times. Histograms are plotted by grouping points into 20 bins on the y-axis.

In UN scenario, Figure 5.8(a) shows the effect of the CONSTANT delaying function on consumers' waiting time. Two main conclusions can be drawn from this figure: (*i*) the order in which consumers arrive has a strong positive correlation with their waiting time, and (*ii*) the distribution of waiting times is denser in the lowest values (between 0 and 20 minutes). Similar results are shown for the LINEAR function in Figure 5.8(b). Although the histogram shows largest bars in the middle of the y-axis, the highest waiting time is around 110 minutes, whereas in the CONSTANT function it is around 170. It is important to notice that the distribution of waiting times is different for each delaying function, but the respective $\mathcal{W}$ metrics are almost equal.

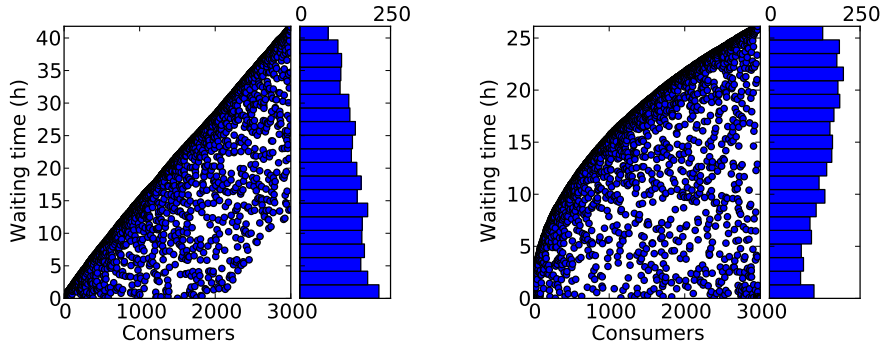
Considering the FC scenario, Figure 5.9(a) shows the effect of the CONSTANT delaying function on consumers' waiting time. Similar to the previous scenario, the size of histogram bars decreases as the waiting time increases. Figure 5.9(b) plots waiting times incurred by the use of the LINEAR function. In this case, the maximum waiting time, $\sim$26 hours, is approximately 62% of the maximum value incurred by the CONSTANT function, $\sim$42 hours.

In order to summarize these four studied scenarios, i.e., UN+CONSTANT, UN+LINEAR,



(a) CONSTANT delaying function (b) LINEAR delaying function

Figure 5.8: Scatter plot and histogram for consumers' waiting time (UN scenario)



(a) CONSTANT delaying function (b) LINEAR delaying function

Figure 5.9: Scatter plot and histogram for consumers' waiting time (FC scenario)

FC+CONSTANT, and FC+LINEAR, values of the average, median, and standard deviation are shown for different proportions of attackers in Table 5.4. Values for the UN scenarios are presented in minutes, whereas FC scenarios are in hours. As one may notice, the 'med' values are lower than corresponding 'avg' for all scenarios, except for FC+LINEAR. This indicates that, for the majority of the cases, 50% of consumers wait less than the $\mathcal{W}$ metric presented earlier.

Table 5.4: Summary of waiting times, presented in minutes for UN and hours for FC, for different proportions of attackers

A	UN+CONSTANT			UN+LINEAR			FC+CONSTANT			FC+LINEAR		
	avg	med	std	avg	med	std	avg	med	std	avg	med	std
0%	30.0	21.9	26.5	30.0	26.6	20.3	11.5	10.3	8.1	11.6	12.2	6.1
10%	41.1	35.5	31.2	34.6	32.3	21.9	13.4	12.0	9.0	12.4	13.0	6.4
30%	63.7	60.8	43.2	44.2	43.7	25.0	18.4	17.4	11.7	14.6	15.3	7.2

The impact of studied attacks on $\mathcal{W}$ can be measured by comparing the 'avg' columns for different values of A. Results show that the CONSTANT delaying function is more sensitive to the number of attackers, i.e., when there are 30% of attackers, 'avg' increases from 30.0 to 63.7 (112%) and from 11.5 to 18.4 (60%) in UN and FC cases, respectively. In contrast, the LINEAR function presents a more robust behavior. In this case, 'avg' increases from 30.0 to 44.2 (47%) and from 11.6 to 14.6 (25%) in UN and FC

cases, respectively. The performance of these two delaying functions is more similar in the presence of 10% of attackers, i.e., 'avg' increases 37% (UN) and 16% (FC) for the CONSTANT function, and 15% (UN) and 6% (FC) for the LINEAR function.

Summary. This analysis showed that users' waiting time is more concentrated in lower values for the CONSTANT function, considering both UN and FC scenarios. However, when employing the LINEAR function, the highest waiting time is ∼35% lower compared to the corresponding CONSTANT function in both UN (from 170 to 110 minutes) and FC (from 40 to 25) scenarios. Table 5.4 indicates that the LINEAR delaying function presents lower overhead in most of the cases. Furthermore, it is possible to observe that the LINEAR function is more robust against massive attacks.

5.4 Considerations on the proposed solution

As shown in the previous sections, the presented solution provides a flexible model to protect consumers from malicious interferences in the system. The strategy behaves proactively to mitigate massive joins of attackers in early stages of the content distribution process. Furthermore, it can be instantiated in different domains and adapted to behave more or less conservatively, leveraging distinct delaying functions. This subsection presents considerations regarding deployment and operation of the conservative strategy in real systems.

This chapter formalized massive attacks and discussed the feasibility of using delaying strategies to improve users' QoE. Although the efficacy of the proposed approach has been shown, a number of practical issues must be addressed in order to enable the integration of the strategy to a real CDS. First, it is necessary to estimate the arrival rate of consumers. Second, since the strategy may behave more or less conservatively, it is also necessary to define the average waiting time to be imposed to consumers. Third, the shape of the delaying function must be chosen, e.g., CONSTANT or LINEAR.

In regard to the estimation of arrival rates, there are several studies in the literature that show user's behavior in file sharing (GUO et al., 2005; ANDRADE et al., 2009) and streaming systems (GILL et al., 2008; CHATZOPOULOU; SHENG; FALOUTSOS, 2010), as representative examples of CDS. These methodologies could be applied to estimate user's arrival rates and used as input for the model presented in this chapter.

Another important discussion is the determination of the average waiting time to be imposed to users, since system administrators need to define proper values for $\mathcal{W}$ and derive the respective delaying function to be employed. The sensitivity analysis presented in Figures 5.2 and 5.4 may support administrators decision. The higher the average waiting time is, the higher QoE users have. It is important to emphasize that there is a trade-off between these metrics and some heuristic is needed to determine proper artificial delays. A simple approach to tackle this would be to use $\mathcal{W}$ so that the derivative of $\mathcal{Q}_{norm}$ with respect to $\mathcal{W}$ is equal to (or lower than) 1, i.e., $\mathcal{W}$ begins to increase equal to (or faster than) $\mathcal{Q}_{norm}$. However, a perfect value for the average waiting time does not exist and an ideal value depends on the nature of the system. Considering critical systems with time constraints, this task is even harder and may compromise the use of delaying strategies. In some cases, depending on content popularity, short delays are enough to mitigate massive attacks. The relationship between content popularity and effectiveness of delaying strategies was previously discussed in this chapter.

In respect to the choice of the delaying function, this chapter presented and discussed two possibilities: CONSTANT and LINEAR. The former allows users to join following a

constant (uniform) rate. This may suit systems whose administrators want to avoid flash crowd behaviors. The latter allows an increasing number of users to join the system. In this case, the strategy may act more conservative in the beginning and less so as time passes. These two intuitive functions are taken as case study in this work, but other strategies (e.g., logarithmic and exponential) may be employed.

One last discussion about the proposed solution is related to the meaning of QoE and the definition introduced, with the $\mathcal{Q}$ metric. This metric captures the rationale behind the health of a system and employs a simple (and easy to be calculated) equation to measure the QoE. For instance, if $\mathcal{Q}$ is 1, one may conclude that every user joined before attackers. Conversely, a $\mathcal{Q}$ equals to 0 means that only attackers have joined the system. This example shows that the $\mathcal{Q}$ metric may not capture important facets of QoE, for example, in scenarios where attackers join later and still can damage the system. However, the metric is proposed as a complement, not a substitute, for other QoE metrics.

5.5 Summary

This chapter presented a flexible model to study the impact of massive attacks on CDS. Two metrics, $\mathcal{Q}$ and $\mathcal{W}$, were introduced and used to instantiate a conservative strategy. Results underscored that slowing down users, even when short waiting times are used, is a prominent solution to speed up the matching between users and expected contents. In the absence of such a mechanism, every attacker has access to the system in the beginning and may act maliciously. However, when the mechanism is turned on, the impact of massive attacks is "diluted" because consumers also arrive and contend with attackers for a chance to join the system. Even in extreme cases with 30% of malicious users arriving *en masse* in the beginning, the majority of consumers are allowed to join in the presence of few attackers.

The main advantages of the proposed solution are threefold. First, unlike most of the previous solutions, the conservative strategy neither relies on user's feedback nor identity management mechanisms. Second, the general massive attack can be divided into smaller instances and the solution may also be applied in distributed environments with minor efforts. Third, since the proposed strategy can behave more or less restrictively, its instantiation may happen in different contexts, such as file sharing and streaming systems.

6 SUMMARY, CONCLUSIONS, AND FUTURE WORK

The goals of this chapter are to summarize the thesis, present the conclusions, and discuss future work. First, the list of incremental steps to support the hypothesis in this thesis is presented. Second, the main contributions are summarized and discussed. Third, and last, prospective directions for future research are shown.

6.1 Summary of contributions

In this thesis, we presented the incremental steps carried out to defend our hypothesis: **"delaying user's actions mitigates massive attacks and improves users' quality of experience."** Based on an evolutive process towards a generic model to represent user's behaviors, a conservative strategy was designed and instantiated in multiple use cases. In a first iteration, we faced some challenges to design a robust strategy based on binary votes to fight content pollution in BitTorrent communities. In a second iteration, this strategy was improved to allow users to express their opinions through annotations, instead of only assigning positive and negative votes. Based on prominent results achieved using conservative strategies, in a third iteration, we generalized the proposed models to increase their applicability and allow further investigations about delaying functions. The complete solution is generic enough to be instantiated in any kind of CDS, but in this thesis we always refer to the two most representative examples, e.g., file sharing and streaming systems. It is important to notice that since delaying functions are employed to mitigate massive attacks, users from some systems may not tolerate long waiting times. In those cases, short waiting times can be considered and still produce reasonable results, as shown in Chapter 5. The three main contributions of this thesis and a brief summary follow.

First iteration. Design and instantiation of a conservative strategy based on binary votes to control the dissemination of potential polluted content during early stages of its publication.

Summary. In this iteration, we have demonstrated the efficiency of the proposed solution, which has imposed significant delays to the dissemination of polluted contents, whereas causing a relatively low delay for the dissemination of non-polluted, legitimate ones. Furthermore, low dissemination delays experienced by non-polluted copies have been observed even when there is a large number of users colluding to hamper the dissemination. These observations showed that the conservative approach proposed in this thesis not only mitigated the dissemination of polluted content, but also required a significant effort to adversely manipulate the mechanism.

Second iteration. Proposal of a flexible mechanism to describe contents using annotations.

Summary. By moving beyond content polarization, we have introduced a new dimension to the content pollution problem. Contents started being considered mislabeled instead of polluted. In the absence of binary votes to build content reputation, we designed a variation metric to detect whether contents are properly described. Results showed that delaying users' downloads enabled a more precise construction of contents vocabulary. As download requests were gradually authorized, attacks *en masse* were thwarted by our solution; further, regular users also arrived and had a chance to download contents.

Third iteration. Generalization of the content pollution attack to a massive attack, and development of a generic model to mitigate these attacks in content distribution systems.

Summary. After analyzing the behavior of our previous approaches (first and second iterations), we realized that the main benefits of using conservative strategies were due to their delaying effect. The impact of massive attacks is reduced because consumers also arrive and contend with attackers for a chance to join. Results showed that, depending on the nature of the attack, it is possible to reduce its impact even imposing small waiting times to users.

Given the aforementioned evolution of this thesis, highlighting the contributions and lessons learned, the next section closes the thesis with a general discussion about our achievements and presents prospective directions for future research.

6.2 Final remarks

The work presented in this thesis investigated the pollution problem under a different perspective. Previous solutions attempted to observe and gather information in order to react against pollution. Those solutions may be very effective in the medium and long run, building a robust base of knowledge about users and contents. However, in short term, massive user joins with a few polluters may be enough to compromise a CDS. Users who download polluted content broadcast it to all their neighbors, producing a chain effect in the pollution dissemination (KUMAR et al., 2006; LEE et al., 2006). For example, previous solutions that try to detect and isolate fake content will allow several users to download polluted content until the first vote is assigned. This may take a while, specially for large files. We argue that our proactive approach is the main factor responsible for reducing pollution propagation and diminishing massive attacks in CDS. Our findings may support system architects to extend our model and instantiate a complete solution in different contexts.

In conclusion, the overall work presented in this thesis underscored the importance of adopting security mechanisms to mitigate malicious behavior in CDS. In the absence of countermeasure mechanisms, we showed that even a small proportion (10%) of attackers was able to subvert the system. The adoption of a conservative strategy demonstrated the efficacy of delaying users in circumventing massive attacks. The simple, yet powerful, strategy managed to thwart attacks *en masse* and protect users against malicious interferences. The user's quality of experience was analyzed and results suggested that there is a

trade-off between delaying users and their resulting experience. Additional investigation should include a survey in order to demonstrate that the proportion of attackers in the system affects the quality of experience observed by users. However, the current work presented in this thesis pushes the state of the art by investigating the content pollution problem and massive attacks under a different perspective. We focused on controlling the impact of massive attacks during early stages on systems' life cycle, which is a novel approach in the context of CDS.

The investigation performed during this thesis leads to the identification of further opportunities of research. First, to demonstrate the effectiveness of conservative strategies compared to traditional solutions, it is desirable to design a methodology for comparison and evaluate them in multiple scenarios. Second, an additional mechanism could be attached to our solution to aggregate information about users and improve overall efficacy. For example, a reputation mechanism in which users evaluate each other would provide a valuable information to any conservative strategy. Third, to have a more dynamic solution, it would be desirable to carry out a deeper investigation on how to tune system parameters for different contexts, preferably in an adaptive manner. Fourth, although we evaluated pieces of our solution using experiments in a controlled environment and others using simulations, it would be desirable to instantiate a complete solution in a real CDS. Discussions on how to do that are presented in the previous chapters of this thesis. It is important to emphasize that these research opportunities could be identified due to the knowhow acquired during the development of this work.

APPENDIX A CAPÍTULO EM PORTUGUÊS

A Internet tem se tornado uma plataforma importante para interação e compartilhamento de conteúdo. Esses conteúdos tipicamente consistem de arquivos pessoais ou de terceiros, que são publicados e recuperados pelos usuários com base em seus próprios interesses (ANDRADE et al., 2005). Relatórios recentes indicaram grandes quantidades conteúdos sendo compartilhados na Internet, em especial de conteúdos multimedia, que representaram um volume representativo de tráfego entre provedores de Internet (GERBER; DOVERSPIKE, 2011). Por exemplo, a quantidade de conteúdo enviado ao YouTube dentro de um período de 60 dias é equivalente ao que teria sido transmitido, sem interrupções, pela NBC, CBS e ABC juntas (The New York Times, 2010; FIGUEIREDO; BENEVENUTO; ALMEIDA, 2011). Para satisfazer essa demanda por serviços eficientes, *sistemas de distribuição de conteúdo* (CDS) precisam ser criados e mantidos (AGER et al., 2011).

No contexto desta tese, CDS são definidos como sistemas usados para compartilhar qualquer tipo de conteúdo na Internet. Duas categorias de CDS se destacam como as mais populares: compartilhamento de arquivos e sistemas de mídia contínua. Sistemas de compartilhamento de arquivos são usados para armazenar e disseminar qualquer tipo de conteúdo, de documentos e imagens a áudios e vídeos. Tendo em vista que arquivos são acessados apenas após serem recuperados totalmente da rede, não existem restrições em relação à ordem em que seus *bytes* são obtidos. Por outro lado, nos sistemas de mídia contínua, os *bytes* precisam ser recuperados de forma sistemática para permitir sua rápida interpretação e reprodução. Nos casos em que o conteúdo multimedia é disseminado enquanto gravado, sem armazenamento prévio, o sistema de distribuição é classificado como "ao vivo". Tais sistemas demandam conexões de rede eficientes para disseminar conteúdos de alta qualidade na Internet.

Arquiteturas par-a-par (P2P) surgiram como potenciais soluções para o aprimoramento da disseminação de conteúdo nos CDS (COHEN, 2003; ZHANG; LIU; PETER YUM, 2005; LIAO et al., 2006; HECHT et al., 2011). Essas arquiteturas viabilizam o balanceamento da carga concentrada nos servidores. Em consequência disso, os usuários (*peers*) assumem parte da tarefa anteriormente exercida pelos servidores. Ao invés de manter entidades centralizadoras para distribuir conteúdos, sistemas P2P são criados para que *peers* interajam para compartilhar dados e maximizar a utilização dos seus recursos. Nesse contexto, a popularização das arquiteturas P2P motivou a comunidade científica a investigar alguns aspectos de pesquisa desafiadores, e.g., otimização de topologias de redes (CHOFFNES; BUSTAMANTE, 2008; CHOFFNES; SÁNCHEZ; BUSTAMANTE, 2010; POESE et al., 2010; CAPOTA et al., 2011), mecanismos de inicialização de sistemas (TADDIA; MAZZINI, 2008) e serviços de descoberta de recursos (IAMNITCHI; FOSTER, 2001). Um desafio com interesse especial a esta tese diz respeito a mecanismos

para conciliar a preferência dos usuários aos conteúdos publicados.

A eficácia dos CDS depende da experiência dos publicadores para descrever adequadamente os conteúdos, aspecto importante para garantir uma boa *qualidade de experiência* (QoE) aos usuários dos sistemas. Uma vez que podem existir divergências entre opiniões de usuários, conteúdos similares podem ser classificados com descrições pouco relacionadas. Além disso, usuários maliciosos podem publicar conteúdos com descrições populares para desviar a atenção dos usuários comuns, atrapalhar suas consultas e/ou promover propagandas indesejadas. Evidências concretas e medições sobre "poluição" têm sido relatadas por administradores de CDS (e.g., comunidades BitTorrent (TORRENTFREAK, 2009; CUEVAS et al., 2010) e rede KaZaa (LIANG et al., 2005)). Essas evidências são reforçadas por um estudo prévio (SANTOS et al., 2011) e resultados de pesquisas que nós conduzimos junto a administradores de comunidades BitTorrent, que reportaram que cerca de 25% de todo conteúdo publicado é poluído e precisa de intervenção manual.

Esforços de pesquisa constantes têm sido feitos para combater poluição de conteúdo em CDS (CAI et al., 2009; ANAND; BHASKAR, 2011; SHIN; REEVES, 2012). Abordagens buscam construir uma base de conhecimento sobre poluidores e conteúdos poluídos para identificar e isolar conteúdos suspeitos depois que eles são publicados. Entretanto, o tempo de reação dessas abordagens até considerar um conteúdo poluído é consideravelmente longo, permitindo uma ampla disseminação de poluição. Além disso, algumas abordagens anteriores buscam polarizar conteúdos entre poluídos ou não, desconsiderando a intrínseca *subjetividade* acerca da classificação dos conteúdos compartilhados.

O objetivo principal desta tese é propor um mecanismo para prover uma boa QoE aos usuários – agindo proativamente durante as fases iniciais da publicação dos conteúdos – e reduzir os efeitos de interferências maliciosas. Para alcançar tal objetivo, três passos principais guiaram o trabalho de pesquisa apresentado nesta tese. Primeiro, propusemos uma estratégia inovadora que opera de forma conservadora para conter a disseminação de poluição. Segundo, estendemos nossa solução para lidar com a subjetividade acerca das descrições dos conteúdos. Terceiro, tratamos o ataque de poluição como um ataque massivo. Para avaliar a solução, experimentos foram executados utilizando testes reais e simulações. Resultados ressaltaram a importância de adotar medidas de segurança para combater comportamentos maliciosos em CDS. Na ausência de mecanismos de contramedida, pequenas proporções (10%) de atacantes foram capazes de comprometer o sistema. A instanciação da estratégia conservadora proposta nesta tese demonstrou a eficácia em atrasar usuários para contornar ataques massivos.

REFERENCES

ADAR, E.; HUBERMAN, B. A. Free Riding on Gnutella. **First Monday**, [S.l.], v.5, n.10, October 2000.

AGER, B. et al. Web content cartography. In: ACM SIGCOMM CONFERENCE ON INTERNET MEASUREMENT (IMC 2011), 11., New York, NY, USA. **Anais...** ACM, 2011. p.585–600.

AL-HAMRA, A.; LEGOUT, A.; BARAKAT, C. **Understanding the Properties of the BitTorrent Overlay**. [S.l.: s.n.], 2007.

ANAGNOSTAKIS, K. G. et al. **On the Impact of Practical P2P Incentive Mechanisms on User Behavior**. [S.l.]: NET Institute, 2006.

ANAND, P. M. R.; BHASKAR, V. Polluted content prevention in Peer-to-Peer file sharing networks. In: ANNUAL IEEE INDIA CONFERENCE (INDICON). **Anais...** [S.l.: s.n.], 2011. p.1–4.

ANDRADE, N. et al. Influences on cooperation in BitTorrent communities. In: ACM SIGCOMM WORKSHOP ON ECONOMICS OF PEER-TO-PEER SYSTEMS (P2PECON 2005), New York, NY, USA. **Anais...** ACM, 2005. p.111–115.

ANDRADE, N. et al. On the Efficiency and Cost of Introducing QoS in BitTorrent. In: IEEE INTERNATIONAL SYMPOSIUM ON CLUSTER COMPUTING AND THE GRID (CCGRID 2007), 7., Washington, DC, USA. **Anais...** IEEE Computer Society, 2007. p.767–772. (CCGrid 2007).

ANDRADE, N. et al. Resource demand and supply in BitTorrent content-sharing communities. **Computer Networks: The International Journal of Computer and Telecommunications Networking**, New York, NY, USA, v.53, n.4, p.515–527, 2009.

ANDRADE, N.; SANTOS-NETO, E.; BRASILEIRO, F. Scalable Resource Annotation in Peer-to-Peer Grids. In: IEEE INTERNATIONAL CONFERENCE ON PEER-TO-PEER COMPUTING (P2P 2008), 8. **Anais...** IEEE Computer Society, 2008. p.231–234.

ANDROUTSELLIS-THEOTOKIS, S. **A Survey of Peer-to-Peer File Sharing Technologies**. 2002.

AWERBUCH, B.; SCHEIDELER, C. Group Spreading: a protocol for provably secure distributed name service. In: AUTOMATA, LANGUAGES AND PROGRAMMING. **Anais...** Springer Berlin / Heidelberg, 2004. p.183–195. (Lecture Notes in Computer Science, v.3142).

BARCELLOS, M. P. et al. Protecting BitTorrent: design and evaluation of effective countermeasures against dos attacks. In: INTERNATIONAL SYMPOSIUM ON RELIABLE DISTRIBUTED SYSTEMS (SRDS 2008), 27., Napoli, Italy. **Anais...** IEEE, 2008. p.73–82.

BENEVENUTO, F. et al. Identifying video spammers in online social networks. In: INTERNATIONAL WORKSHOP ON ADVERSARIAL INFORMATION RETRIEVAL ON THE WEB (AIRWEB 2008), 4., New York, NY, USA. **Anais...** ACM, 2008. p.45–52.

BENEVENUTO, F. et al. Detecting spammers and content promoters in online video social networks. In: INTERNATIONAL ACM SIGIR CONFERENCE ON RESEARCH AND DEVELOPMENT ON INFORMATION RETRIEVAL (SIGIR 2009), 32., New York, NY, USA. **Anais...** ACM, 2009. p.620–627. (SIGIR 2009).

BENEVENUTO, F. et al. Practical Detection of Spammers and Content Promoters in Online Video Sharing Systems. **IEEE Transactions on Systems, Man, and Cybernetics (Part B: Cybernetics)**, [S.l.], v.42, n.3, p.688–701, june 2012.

BIAN, J. et al. A few bad votes too many?: towards robust ranking in social media. In: INTERNATIONAL WORKSHOP ON ADVERSARIAL INFORMATION RETRIEVAL ON THE WEB (AIRWEB 2008), 4., New York, NY, USA. **Anais...** ACM, 2008. p.53–60.

BITSNOOP. **Bitsnoop P2P Search**. 2012.

BLOND, S. L. et al. One Bad Apple Spoils the Bunch: exploiting p2p applications to trace and profile tor users. In: USENIX WORKSHOP ON LARGE-SCALE EXPLOITS AND EMERGENT THREATS (LEET 2011), 4., Boston, United States. **Anais...** USENIX, 2011.

BLOND, S. L. et al. I Know Where You are and What You are Sharing: exploiting p2p communications to invade users' privacy. In: ACM SIGCOMM CONFERENCE ON INTERNET MEASUREMENT (IMC 2011), 11. **Anais...** ACM/USENIX, 2011.

BOXIFY. **boxify.me - simple file-sharing for groups**. 2012.

CAI, Z. et al. A holistic mechanism against file pollution in peer-to-peer networks. In: ACM SYMPOSIUM ON APPLIED COMPUTING (SAC 2009), 24., New York, NY, USA. **Anais...** ACM, 2009. p.28–34.

CAPOTA, M. et al. Inter-swarm resource allocation in BitTorrent communities. In: IEEE INTERNATIONAL CONFERENCE ON PEER-TO-PEER COMPUTING (P2P 2011), 11. **Anais...** [S.l.: s.n.], 2011. p.300–309.

CASTRO, M. et al. SplitStream: high-bandwidth multicast in cooperative environments. In: ACM SYMPOSIUM ON OPERATING SYSTEMS PRINCIPLES (SOSP 2003), 19., New York, NY, USA. **Anais...** ACM, 2003. p.298–313. (SOSP 2003).

CHATZOPOULOU, G.; SHENG, C.; FALOUTSOS, M. A First Step Towards Understanding Popularity in YouTube. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER COMMUNICATIONS WORKSHOPS (INFOCOM WKSHPS 2010), 29. **Anais...** [S.l.: s.n.], 2010. p.1–6.

CHOFFNES, D. R.; BUSTAMANTE, F. E. Taming the torrent: a practical approach to reducing cross-isp traffic in peer-to-peer systems. In: ACM SIGCOMM CONFERENCE ON DATA COMMUNICATION (SIGCOMM 2008). **Anais...** ACM, 2008. p.363–374.

CHOFFNES, D. R.; SÁNCHEZ, M. A.; BUSTAMANTE, F. E. Network positioning from the edge: an empirical study of the effectiveness of network positioning in p2p systems. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER COMMUNICATIONS (INFOCOM 2010), 29., Piscataway, NJ, USA. **Anais...** IEEE, 2010. p.291–295.

COHEN, B. Incentives Build Robustness in BitTorrent. In: WORKSHOP ON ECONOMICS OF PEER-TO-PEER SYSTEMS (P2PECON 2003), Berkeley, CA, USA. **Anais...** [S.l.: s.n.], 2003.

COSTA, C.; ALMEIDA, J. Reputation Systems for Fighting Pollution in Peer-to-Peer File Sharing Systems. In: IEEE INTERNATIONAL CONFERENCE ON PEER-TO-PEER COMPUTING (P2P 2007), 7. **Anais...** IEEE, 2007. p.53–60.

COSTA, C. et al. Fighting pollution dissemination in peer-to-peer networks. In: ACM SYMPOSIUM ON APPLIED COMPUTING (SAC 2007), 22., New York, NY, USA. **Anais...** ACM, 2007. p.1586–1590.

CUEVAS, R. et al. Is content publishing in BitTorrent altruistic or profit-driven? In: INTERNATIONAL CONFERENCE ON EMERGING NETWORKING EXPERIMENTS AND TECHNOLOGIES (CONEXT 2010), 6. **Anais...** ACM, 2010. p.11:1–11:12.

DALE, C. et al. Evolution and Enhancement of BitTorrent Network Topologies. In: INTERNATIONAL WORKSHOP ON QUALITY OF SERVICE (IWQOS 2008), 16. **Anais...** [S.l.: s.n.], 2008. p.1–10.

DAMIANI, E. et al. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In: ACM CONFERENCE ON COMPUTER AND COMMUNICATIONS SECURITY (CCS 2002), 9., New York, NY, USA. **Anais...** ACM, 2002. p.207–216.

DHUNGEL, P. et al. A Measurement Study of Attacks on BitTorrent Leechers. In: INTERNATIONAL WORKSHOP ON PEER-TO-PEER SYSTEMS (IPTPS 2008), 7. **Anais...** [S.l.: s.n.], 2008.

DHUNGEL, P. et al. A Measurement Study of Attacks on BitTorrent Seeds. In: IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS (ICC 2011). **Anais...** [S.l.: s.n.], 2011. p.1–5.

DOUCEUR, J. R. The Sybil Attack. In: INTERNATIONAL WORKSHOP ON PEER-TO-PEER SYSTEMS (IPTPS 2002), 1. **Anais...** [S.l.: s.n.], 2002. p.251–260.

FAUZIE, M. D. et al. A temporal view of the topology of dynamic Bittorrent swarms. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER COMMUNICATIONS WORKSHOPS (INFOCOM WKSHPS 2011), 30. **Anais...** [S.l.: s.n.], 2011. p.894–899.

FELDMAN, M. et al. Free-riding and whitewashing in peer-to-peer systems. **IEEE Journal on Selected Areas in Communications**, [S.l.], v.24, n.5, p.1010–1019, 2006.

FIGUEIREDO, F.; BENEVENUTO, F.; ALMEIDA, J. M. The tube over time: characterizing popularity growth of youtube videos. In: ACM INTERNATIONAL CONFERENCE ON WEB SEARCH AND DATA MINING (WSDM 2011), 4., New York, NY, USA. **Anais...** ACM, 2011. p.745–754.

FURNAS, G. W. et al. Why do tagging systems work? In: EXTENDED ABSTRACTS ON HUMAN FACTORS IN COMPUTING SYSTEMS (CHI 2006), New York, NY, USA. **Anais...** ACM, 2006. p.36–39. (CHI EA 2006).

GERBER, A.; DOVERSPIKE, R. Traffic types and growth in backbone networks. In: OPTICAL FIBER COMMUNICATION CONFERENCE AND EXPOSITION, AND NATIONAL FIBER OPTIC ENGINEERS CONFERENCE (OFC/NFOEC 2011). **Anais...** [S.l.: s.n.], 2011. p.1–3.

GILL, P. et al. Characterizing User Sessions on YouTube. In: SPIE/ACM MULTIMEDIA COMPUTING AND NETWORKING (MMCN 2008), 15. **Anais...** ACM, 2008.

GOLDER, S. A.; HUBERMAN, B. A. Usage patterns of collaborative tagging systems. **Journal of Information Science**, Thousand Oaks, CA, USA, v.32, n.2, p.198–208, 2006.

GUO, L. et al. Measurements, analysis, and modeling of BitTorrent-like systems. In: CONFERENCE ON INTERNET MEASUREMENT (IMC 2005), 5., Berkeley, CA, USA. **Anais...** USENIX Association, 2005. p.35–48.

HARIDASAN, M.; RENESSE, R. van. Defense Against Intrusion in a Live Streaming Multicast System. In: IEEE INTERNATIONAL CONFERENCE ON PEER-TO-PEER COMPUTING (P2P 2006), 6., Washington, DC, USA. **Anais...** IEEE Computer Society, 2006. p.185–192.

HECHT, F. V. et al. LiveShift: mesh-pull live and time-shifted p2p video streaming. In: IEEE CONFERENCE ON LOCAL COMPUTER NETWORKS (LCN 2011), 36. **Anais...** [S.l.: s.n.], 2011. p.315–323.

HECKNER, M.; NEUBAUER, T.; WOLFF, C. Tree, funny, toread, google: what are tags supposed to achieve? a comparative analysis of user keywords for different digital resource types. In: ACM WORKSHOP ON SEARCH IN SOCIAL MEDIA (SSM 2008), New York, NY, USA. **Anais...** ACM, 2008. p.3–10.

HEI, X. et al. A Measurement Study of a Large-Scale P2P IPTV System. **IEEE Transactions on Multimedia**, [S.l.], v.9, n.8, p.1672–1687, 2007.

HUANG, Z. et al. CloudStream: delivering high-quality streaming videos through a cloud-based svc proxy. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER COMMUNICATIONS (INFOCOM 2011), 30. **Anais...** [S.l.: s.n.], 2011. p.201–205.

HUGHES, D.; COULSON, G.; WALKERDINE, J. Free riding on Gnutella revisited: the bell tolls? **IEEE Distributed Systems Online**, [S.l.], v.6, n.6, p.1–18, june 2005.

HUSNA, H.; PHITHAKKITNUKOON, S.; DANTU, R. Traffic Shaping of Spam Botnets. In: IEEE CONSUMER COMMUNICATIONS AND NETWORKING CONFERENCE (CCNC 2008), 5. **Anais...** [S.l.: s.n.], 2008. p.786–787.

IAMNITCHI, A.; FOSTER, I. On Fully Decentralized Resource Discovery in Grid Environments. In: LEE, C. (Ed.). **Grid Computing (GRID 2001)**. [S.l.]: Springer Berlin / Heidelberg, 2001. p.51–62. (Lecture Notes in Computer Science, v.2242).

IMDB. **The Internet Movie Database**. 2012.

JIA, J.; LI, C.; CHEN, C. Characterizing PPStream across Internet. In: IFIP INTERNATIONAL CONFERENCE ON NETWORK AND PARALLEL COMPUTING (NPC 2007 – WORKSHOPS). **Anais...** [S.l.: s.n.], 2007. p.413–418.

JOARDER, A. H.; LATIF, R. M. Standard Deviation for Small Samples. **Teaching Statistics**, [S.l.], v.28, n.2, p.40–43, 2006.

JOSANG, A.; ISMAIL, R.; BOYD, C. A Survey of Trust and Reputation Systems for Online Service Provision. In: **Elsevier Decision Support Systems**. [S.l.]: Elsevier, 2007.

JOSANG, A.; POPE, S.; HAYWARD, R. Trust Network Analysis with Subjective Logic. In: AUSTRALASIAN COMPUTER SCIENCE CONFERENCE (ACSC 2006), 29., Darlinghurst, Australia, Australia. **Anais...** Australian Computer Society: Inc., 2006. p.85–94.

JUSTBEAMIT. **JustBeamIt - file transfer made easy**. 2012.

KAMVAR, S. D.; SCHLOSSER, M. T.; GARCIA-MOLINA, H. The Eigentrust algorithm for reputation management in P2P networks. In: INTERNATIONAL CONFERENCE ON WORLD WIDE WEB (WWW 2003), 12., New York, NY, USA. **Anais...** ACM, 2003. p.640–651.

KONRATH, M. A.; BARCELLOS, M. P.; MANSILHA, R. B. Attacking a Swarm with a Band of Liars: evaluating the impact of attacks on bittorrent. In: IEEE INTERNATIONAL CONFERENCE ON PEER-TO-PEER COMPUTING (P2P 2007), 7. **Anais...** [S.l.: s.n.], 2007. p.37–44.

KOUTRIKA, G. et al. Combating spam in tagging systems: an evaluation. **ACM Transactions on the Web**, New York, NY, USA, v.2, n.4, p.1–34, 2008.

KREITZ, G.; NIEMELA, F. Spotify – Large Scale, Low Latency, P2P Music-on-Demand Streaming. In: IEEE INTERNATIONAL CONFERENCE ON PEER-TO-PEER COMPUTING (P2P 2010), 10. **Anais...** [S.l.: s.n.], 2010. p.1–10.

KRYCZKA, M. et al. Unrevealing the structure of live BitTorrent swarms: methodology and analysis. In: IEEE INTERNATIONAL CONFERENCE ON PEER-TO-PEER COMPUTING (P2P 2011), 11. **Anais...** [S.l.: s.n.], 2011. p.230–239.

KUMAR, R. et al. Fluid Modeling of Pollution Proliferation in P2P Networks. **ACM SIGMetrics Performance Evaluation Review**, [S.l.], v.34, n.1, p.335–346, 2006.

LEE, U. et al. Understanding Pollution Dynamics in P2P File Sharing. In: INTERNATIONAL WORKSHOP ON PEER-TO-PEER SYSTEMS (IPTPS 2006), 5. **Anais...** [S.l.: s.n.], 2006.

LEGOUT, A.; URVOY-KELLER, G.; MICHIARDI, P. Rarest first and choke algorithms are enough. In: ACM SIGCOMM CONFERENCE ON INTERNET MEASUREMENT (IMC 2006), 6., New York, NY, USA. **Anais...** ACM, 2006. p.203–216.

LEHMANN, M. B. et al. Denial-of-service attacks and countermeasures on BitTorrent. **Computer Networks: The International Journal of Computer and Telecommunications Networking**, [S.l.], v.56, n.15, p.3479–3498, october 2012.

LI, K.; PU, C.; AHAMAD, M. Resisting Spam Delivery by TCP Damping. In: CONFERENCE ON EMAIL AND ANTI-SPAM (CEAS 2004), 1. **Anais...** [S.l.: s.n.], 2004.

LIANG, J. et al. Pollution in P2P File Sharing Systems. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER COMMUNICATIONS (INFOCOM 2005), 24., Miami, Florida, USA. **Anais...** [S.l.: s.n.], 2005. p.1586–1590.

LIAO, X. et al. AnySee: peer-to-peer live streaming. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER COMMUNICATIONS (INFOCOM 2006), 25. **Anais...** [S.l.: s.n.], 2006. p.1–10.

LIU, Y.; GUO, Y.; LIANG, C. A survey on peer-to-peer video streaming systems. **Peer-to-Peer Networking and Applications**, [S.l.], v.1, p.18–28, 2008.

LIU, Y.; YANG, Y.; SUN, Y. L. Detection of collusion behaviors in online reputation systems. In: ASILOMAR CONFERENCE ON SIGNALS, SYSTEMS AND COMPUTERS (ASILOMAR 2008), 42. **Anais...** [S.l.: s.n.], 2008. p.1368–1372.

LLETÍ, R. et al. Selecting variables for k-means cluster analysis by using a genetic algorithm that optimises the silhouettes. **Analytica Chimica Acta**, [S.l.], v.515, n.1, p.87–100, 2004.

LU, Y. et al. Assessing the Quality of Experience of SopCast. **International Journal of Internet Protocol Technology**, Inderscience Publishers, Geneva, Switzerland, v.4, n.1, p.11–23, March 2009.

MAGHAREI, N.; REJAIE, R. PRIME: peer-to-peer receiver-driven mesh-based streaming. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER COMMUNICATIONS (INFOCOM 2007), 26. **Anais...** [S.l.: s.n.], 2007. p.1415–1423.

MARLOW, C. et al. HT06, tagging paper, taxonomy, Flickr, academic article, to read. In: CONFERENCE ON HYPERTEXT AND HYPERMEDIA (HYPERTEXT 2006), 17., New York, NY, USA. **Anais...** ACM, 2006. p.31–40.

MEIER, R.; WATTENHOFER, R. ALPS: authenticating live peer-to-peer streams. In: INTERNATIONAL SYMPOSIUM ON RELIABLE DISTRIBUTED SYSTEMS (SRDS 2008), 27. **Anais...** [S.l.: s.n.], 2008. p.45–52.

MINUS. **Minus - Share simply**. 2012.

MOL, J. D.; EPEMA, D. H. P.; SIPS, H. J. The Orchard Algorithm: building mulicast trees for p2p video multicasting without free-riding. **IEEE Transactions on Multimedia**, [S.l.], v.9, n.8, p.1593–1604, December 2007.

PC Magazine. **P2P Music File Sharing Dropped After Limewire Shutdown**. 2011.

PIATEK, M. et al. Contracts: practical contribution incentives for p2p live streaming. In: USENIX SYMPOSIUM ON NETWORKED SYSTEMS DESIGN AND IMPLEMENTATION (NSDI 2010, 7., Berkeley, CA, USA. **Anais...** USENIX Association, 2010. (NSDI 2010).

POESE, I. et al. Improving content delivery using provider-aided distance information. In: ACM SIGCOMM CONFERENCE ON INTERNET MEASUREMENT (IMC 2010), 10., New York, NY, USA. **Anais...** ACM, 2010. p.22–34.

POUWELSE, J. A. et al. Tribler: a social-based peer-to-peer system. **Concurrency and Computation: Practice and Experience**, [S.l.], v.20, n.2, p.127–138, 2008.

QIU, D.; SRIKANT, R. Modeling and performance analysis of BitTorrent-like peer-to-peer networks. In: ACM CONFERENCE ON APPLICATIONS, TECHNOLOGIES, ARCHITECTURES, AND PROTOCOLS FOR COMPUTER COMMUNICATIONS (SIGCOMM 2004), New York, NY, USA. **Anais...** ACM, 2004. p.367–378.

ROCHA, A. A. de Aragão et al. On P2P systems for enterprise content delivery. In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUÍDOS (SBRC 2009), 27., Recife, PE, Brazil. **Anais...** [S.l.: s.n.], 2009. p.379–392.

ROINESTAD, H. et al. Incentives for social annotation. In: ACM CONFERENCE ON HYPERTEXT AND HYPERMEDIA (HT 2009), 20., New York, NY, USA. **Anais...** ACM, 2009. p.327–328. (HT 2009).

SANTOS, F. R. et al. Funnel: choking polluters in bittorrent file sharing communities. **IEEE Transactions on Network and Service Management**, [S.l.], v.8, n.4, p.310–321, december 2011.

SANTOS, F. R. et al. Beyond Pollution and Taste: a tag-based strategy to increase download quality in p2p file sharing systems. **Computer Communications: The International Journal for the Computer and Telecommunications Industry**, New York, NY, USA, v.36, n.2, january 2013.

SANTOS, F. R. et al. Slowing Down to Speed Up: mitigating collusion attacks in content distribution systems. In: IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT (IM 2013), 13. **Anais...** IEEE Communications Society, 2013.

SANTOS-NETO, E. et al. Individual and social behavior in tagging systems. In: ACM CONFERENCE ON HYPERTEXT AND HYPERMEDIA (HT 2009), 20., New York, NY, USA. **Anais...** ACM, 2009. p.183–192. (HT 2009).

SANTOS-NETO, E. et al. **Reuse, Temporal Dynamics, Interest Sharing, and Collaboration in Social Tagging Systems**. [S.l.: s.n.], 2013.

SCHMIDT, A. H. et al. Characterizing dissemination of illegal copies of content through monitoring of BitTorrent networks. In: IEEE/IFIP NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM (NOMS 2012), 13. **Anais...** [S.l.: s.n.], 2012. p.327–334.

SCHULZE, H.; MOCHALSKI, K. **Internet Study 2008-2009**. 2009.

SHAH, P.; PARIS, J.-F. cois. Peer-to-Peer Multimedia Streaming Using BitTorrent. In: IEEE INTERNATIONAL PERFORMANCE COMPUTING AND COMMUNICATIONS CONFERENCE (IPCCC 2007), 26. **Anais...** [S.l.: s.n.], 2007. p.340–347.

SHIN, K.; REEVES, D. S. Winnowing: protecting p2p systems against pollution through cooperative index filtering. **Journal of Network and Computer Applications**, [S.l.], v.35, n.1, p.72–84, 2012.

SILVERSTON, T. et al. Traffic analysis of peer-to-peer IPTV communities. **Elsevier Computer Networks**, New York, NY, USA, v.53, n.4, p.470–484, 2009.

SILVERSTON, T.; FOURMAUX, O. P2P IPTV measurement: a case study of tvants. In: INTERNATIONAL CONFERENCE ON EMERGING NETWORKING EXPERIMENTS AND TECHNOLOGIES (CONEXT 2006), 2., New York, NY, USA. **Anais...** ACM, 2006. p.1–2. (CoNEXT 2006).

SINGH, A. et al. Eclipse Attacks on Overlay Networks: threats and defenses. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER COMMUNICATIONS (INFOCOM 2006), 25., Barcelona, Catalunya, Spain. **Anais...** [S.l.: s.n.], 2006. p.1–12.

SO, J.; REEVES, D. Defending against Sybil Nodes in BitTorrent. In: **10th IFIP Networking (Networking 2011)**. [S.l.]: Springer, 2011. p.25–39. (Lecture Notes in Computer Science, v.6641).

SPOTO, S. et al. Analysis of PPLive through active and passive measurements. In: IEEE INTERNATIONAL PARALLEL & DISTRIBUTED PROCESSING SYMPOSIUM (IPDPS 2009). **Anais...** [S.l.: s.n.], 2009. p.1–7.

TADDIA, C.; MAZZINI, G. A multicast-anycast based protocol for trackerless BitTorrent. In: INTERNATIONAL CONFERENCE ON SOFTWARE, TELECOMMUNICATIONS AND COMPUTER NETWORKS (SOFTCOM 2008), 16. **Anais...** [S.l.: s.n.], 2008. p.264 –268.

The New York Times. **Uploading the Avant-Garde**. 2010.

TIMPANARO, J. P. et al. BitTorrent's Mainline DHT Security Assessment. In: IFIP INTERNATIONAL CONFERENCE ON NEW TECHNOLOGIES, MOBILITY AND SECURITY (NTMS 2011), 4. **Anais...** [S.l.: s.n.], 2011. p.1–5.

TORRENTFREAK. **Fake aXXo Torrents Bombard BitTorrent**. 2009.

URDANETA, G.; PIERRE, G.; STEEN, M. V. A survey of DHT security techniques. **ACM Computing Surveys**, New York, NY, USA, v.43, p.8:1–8:49, February 2011.

VIEIRA, A. B. et al. SimplyRep: a simple and effective reputation system to fight pollution in p2p live streaming. **Computer Networks: The International Journal of Computer and Telecommunications Networking**, [S.l.], 2012.

WALSH, K.; SIRER, E. G. Experience with an object reputation system for peer-to-peer filesharing. In: USENIX SYMPOSIUM ON NETWORKED SYSTEMS DESIGN & IMPLEMENTATION (NSDI 2006), 3. **Anais...** USENIX Association, 2006. p.1.

YANG, S. et al. The Content Pollution in Peer-to-Peer Live Streaming Systems: analysis and implications. In: INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING (ICPP 2008), 37., Los Alamitos, CA, USA. **Anais...** IEEE Computer Society, 2008. p.652–659.

ZHANG, X.; LIU, J.; PETER YUM, T. shing. Coolstreaming/donet: a data-driven overlay network for peer-to-peer live media streaming. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER COMMUNICATIONS (INFOCOM 2005), 24. **Anais...** IEEE, 2005. p.2102–2111.