Faculdade de Engenharia da Universidade do Porto

Programa Doutoral em Engenharia Informática (ProDEI)

Mistrustful P2P

Peer-to-Peer File Sharing Model to Hide User Content Interests

---

Pedro Miguel Moreira da Silva

Supervisor: Prof. Manuel Alberto Pereira Ricardo

Ph.D. Thesis

July 6, 2017

Pedro Miguel Moreira da Silva
Mistrustful P2P: Peer-to-Peer File Sharing Model to Hide User Content Interests
Ph.D. Thesis. July 6, 2017

Faculdade de Engenharia da Universidade do Porto
Rua Dr. Roberto Frias
4200-465 Porto

INESC TEC
Campus da FEUP
Rua Dr. Roberto Frias
4200-465 Porto

ABSTRACT

In the last two decades, the advances in computer technology have drastically changed the media landscape, enabling consumers to become also producers and distributors of creative work, something that in the past was mostly limited to professional parties. However, these advances have also enabled organizations and individuals to collect information about users, combine facts from different sources, and merge them to create databases of personal information that were previously impossible to set up.

Peer-to-peer networks endowed individuals with the means to easily and efficiently distribute digital media over the Internet, and have become an important source of personal information because they are extensively used for large-scale file sharing, user content interests may be trivially identified, and the download of contents can be trivially proved. Thus, users seek for privacy-preserving systems to, among others, avoid user profiling, and to privately download legal contents that may be embarrassing or objectionable.

Several privacy-preserving peer-to-peer systems have been proposed to address the user profiling and content interest identification issues, but they still require peers to advertise what they download. Moreover, legitimate and well-intended users may be held legally liable, given that these systems do not address any legal issues that may arise from their use. Lacking alternatives, users have adopted general-purpose anonymity systems for peer-to-peer file sharing, misunderstanding the privacy guarantees provided by such systems, in particular when relaying traffic of insecure applications. Anonymity systems provide a channel to anonymously transmit messages, but the information disclosed by their content cannot be anonymized. Thus, this thesis proposes a novel peer-to-peer file sharing model, the Mistrustful P2P model, to address the user profiling and content interest identification issues.

The Mistrustful P2P model hides user content interests through plausible deniability, has no trust requirements, and prevents user legal liability for legitimate usage while enabling timely downloads. It deterministically hides user content interests against passive attacks of any size, since peers do no not advertise what they download, and against active attacks of a size up to a configured level. Its performance evaluation shows that peers are able to timely download contents, and that, when considering minimum protection, its performance is close to the one of traditional peer-to-peer file sharing systems.

Resumo

Os avanços tecnológicos ocorridos nas duas últimas décadas mudaram drasticamente o panorama dos conteúdos multimédia e permitiram que os consumidores se tornassem também produtores e distribuidores de trabalho criativo, algo que no passado estava limitado quase exclusivamente a profissionais. Contudo, tanto organizações como indivíduos passaram também a ser capazes de recolher informação sobre os utilizadores, de combinar factos de diferentes fontes e de os associar para criar bases de dados de informação pessoal que antes seriam impossíveis de obter.

As redes *peer-to-peer* dotaram os indivíduos com os meios necessários para fácil e eficientemente distribuir conteúdos digitais através da Internet, tornando-se numa importante fonte de informação pessoal porque são amplamente usadas para partilhar conteúdos em larga escala, a identificação dos interesses dos utilizadores é trivial e a transferência de conteúdos pode ser provada trivialmente. Por isso, os utilizadores procuram por soluções de privacidade para, entre outros, evitar a criação de perfis e para transferir de modo privado conteúdos legais mas que possam ser embaraçosos ou questionáveis.

Foram propostas diversas soluções de privacidade, mas os *peers* continuam a ter que anunciar o que descarregam. Mais, utilizadores idóneos e cumpridores da lei podem ser alvo de processos legais dado que não são tratadas quaisquer questões legais decorrentes da sua utilização. Sem alternativas, os utilizadores adoptaram soluções genéricas de anonimato para redes *peer-to-peer*, não compreendendo as garantias de segurança disponibilizadas, em particular quando encaminham tráfego de aplicações inseguras. As soluções de anonimato fornecem um canal para a troca anónima de mensagens, mas a informação revelada pelo conteúdo das mesmas não pode ser tornada anónima. Assim, esta tese propõe um novo modelo para partilha de ficheiros em redes *peer-to-peer*, o modelo Mistrustful P2P, para tratar este problema.

O modelo Mistrustful P2P oculta os interesses dos utilizadores através de negação plausível, não tem requisitos de confiança, evita problemas legais e permite a transferência de conteúdos em tempo útil. Os interesses dos utilizadores são deterministicamente protegidos contra atacantes passivos de qualquer dimensão, dado que os *peers* não anunciam o que descarregam, e contra ataques activos com dimensão até a um nível configurado. A avaliação de desempenho mostra que os *peers* conseguem transferir conteúdos em tempo útil e que, para um nível de protecção mínimo, o seu desempenho é próximo do de uma solução convencional para partilha de ficheiros em redes *peer-to-peer*.

Acknowledgement

This work would have not been possible without the invaluable contribute and support of numerous people, both at professional and personal levels. Being this a thesis about privacy preservation, it should come as no surprise that I opt to name only my supervisors, although I would like to express my gratitude to all of them. They managed to fix countless typos, ambiguous or less clear sentences, and many other technical and non-technical issues that I kept stubbornly adding throughout the years. I am quite confident that, despite their best efforts, I managed to slip through some of them into the final version.

I would like to express my deepest gratitude to Jaime Dias, who followed my work more closely, and highlight his contribute and his selflessness. He ended up being an informal supervisor, in a certain extent, because he always managed to find time to help me pursuing my Ph.D. even when he could not find time to pursue his.

I would like to thank my supervisors, Prof. Manuel Ricardo and Jaime Dias, for their unconditional belief and support, mainly when I was exploring areas that they were not as knowledgeable about and much less myself, for the countless discussions over these years, for their remarkable guidance, for their spot-on reviews, for their continuous and timely availability, and for many other things that I am unable to recall right now.

At a personal level, I would like to stand out the support and understanding without which the outcome of this work would have not been the same, mainly when I could not be present or forgot something important that I should have not. Lastly, I would like to thank specially for the one person that was always by my side throughout this journey.

Support Funding Acknowledgments

Project "TEC4Growth - Pervasive Intelligence, Enhancers and Proofs of Concept with Industrial Impact/NORTE-01-0145-FEDER-000020" is financed by the North Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, and through the European Regional Development Fund (ERDF).

This work is financed by FEDER through Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme through the Agência Nacional de Inovação (ANI) within the scope of the project no. 3468 (MareCom).

This work is financed by the ERDF – European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme within project «POCI-01-0145-FEDER-006961», and by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia as part of project «UID/EEA/50014/2013».

The author also thanks FCT for the grant under the fellowship SFRH/BD/69388/2010.

> *Perfection is not attainable. But if we chase perfection, we can catch excellence.*
>
> **— Vince Lombardi**

Contents

List of Figures

List of Tables

GLOSSARY

block — An erasure coded chunk. 24–39, 45, 64–66, 68, 70, 74–76, 78, 81, 82, 84, 85

chunk — A partition or piece of a content. 5–7, 14, 15, 21, 24–26, 30, 35, 37–39, 47, 69, 82, 84

colluding peer — A peer working coordinately (in collusion) with one or more peers to launch an attack. 2, 7, 8, 28, 29, 63, 82

collusion attack — An attack in which peers controlled by one or more entities coordinate their efforts. 28, 29, 63, 64

commoner — A peer only willing to download a content if the user's privacy requirements can be met. 26–28, 32, 36, 64, 81, 82

content — Set of one or more files, identified by the hash of its data, and partitioned into one or more chunks. 1, 2, 4–9, 12–15, 21, 23–26, 28–30, 33–36, 38, 39, 45, 47, 63–68, 70, 71, 73–82, 84, 85

content availability — A content is considered to be available if it can be fully downloaded. 6, 13, 84

content interest disguise — Privacy-preserving technique that consists in downloading both genuine and cover contents to disguise user content interests. 8, 12, 13, 23, 25, 26, 32, 35, 36, 63, 78, 79, 81, 82, 84

cover content — A content of no interest to the user that is downloaded, either in part or fully, just to disguise his content interests. 8, 23–25, 27, 30, 33, 35, 36, 65–71, 76, 78, 81, 82, 84

direct liability — Lawful accountability and obligations resulting from one's own acts or omissions. 3, 4, 81

eligible peer — A peer to which block requests can still be sent. 27, 33–35

genuine content — A content that the user is interested in downloading. 8, 23–25, 27, 30, 33, 35, 36, 65, 67, 70, 81, 82, 84

hash — An alphanumeric string used to identify and to verify the integrity of the data being transferred. 5, 7, 79

indirect liability — Lawful accountability and obligations resulting from inducing, contributing to or failing to prevent someone else's acts or omissions. 3, 4, 81

leecher — A peer still downloading a content, and that shares the chunks it has already downloaded. 5, 6

legal liability — Lawful accountability and obligations for one's acts or omissions, either their own (direct liability) or someone else's (indirect liability). 1, 2, 4, 5, 7, 8, 11–15, 21, 23, 24, 35, 36, 63, 66, 67, 77, 79–81, 85

minimum network disguise overhead — The minimum amount of blocks that need to be downloaded per cover content. 8, 24, 29, 63, 69–71, 76, 82, 84

misleading content — A content published with misleading description aiming at driving peers to unknowingly and unwillingly download unethical or illegal contents. 7, 13, 14, 25, 36, 66, 82

ns-3 routing protocol — In the context of ns-3 network simulator, a routing protocol is simply a class or module that provides a route for outgoing packets and is able to forward, if needed, incoming packets. Unlike the canonical definition of routing protocols, which implies periodically and explicitly sending protocol messages, an ns-3 routing protocol may populate the routing tables using only global knowledge of the available network links without sending any protocol messages at all. 8, 9, 11, 18–20, 50–52, 55, 57–61, 67, 80, 83, 85

peer — A network node. 2, 5–9, 12–15, 19, 21–36, 49, 64–71, 73–85

plausible deniability — In the context of this work, it is the ability for a user to deny having any interest in downloading a content, having access to its data, or any responsibility for its full download. 8, 23, 35, 63, 66, 79, 81, 82

privacy — In the context of this work, it is the concealment of user content interests. 7, 8, 23, 24, 26, 27, 29–32, 35, 36, 63, 65, 67, 71, 77, 81, 82, 84, 85

proof of access — Proof of access to content data. 24, 25, 35, 81, 82

proof of download — Proof of full content download. 24, 25, 35, 84

publisher — The entity publishing a content. 28, 29, 64, 65, 81, 82, 84

seeder — For traditional P2P systems, it is a peer that has all data chunks and is just sharing them. For the Mistrustful P2P model, it is a peer interested in distributing a content (e.g., on behalf of an author) that has no privacy requirements. 5, 6, 14, 25, 26, 28, 36, 64, 67, 69–71, 73–76, 78, 81, 82, 84

simulated time — The duration of a given simulation. 69

simulation time — Time required to run a simulation. 19, 58–60, 83

swarm — The set of peers sharing a content. 5–7, 13, 14, 23, 27, 34, 65, 68, 81

Sybil attack — The creation of multiple pseudonymous identities. 28, 29, 63, 64

torrent file — File providing the metadata of a content. 5, 45

tracker — A network node providing lists of peers in swarms by keeping track of which peers are sharing which contents. 5–7, 13, 23, 27, 28, 34, 64, 65, 68, 69, 79, 81

untrusted P2P network — Public and open access P2P network composed of large groups of untrusted peers. 2, 7, 8, 23, 29, 80, 81

user — A person. 1, 2, 4, 5, 7, 8, 11–15, 21, 23–25, 27, 29, 30, 35, 36, 63–67, 70, 77, 79–82

List of Abbreviations

AS	Autonomous System. 49
BEC	Binary Erasure Channel. 37
CIDR	Classless Internet-Domain Routing. 9, 50, 52, 55, 58
CMNT	Complex Mersenne Number Transform. 39, 42, 43, 45–48
CPU	Central Processing Unit. 8, 11, 16–18, 30, 40, 48, 49, 58, 78, 82
DDoS	Distributed Denial-of-Service. 63
DFT	Discrete Fourier Transform. 16
DIF	Decimation-In-Frequency. 42
DIT	Decimation-In-Time. 42
DoS	Denial-of-Service. 63, 64
FEC	Forward Error Correction. 37
FFT	Fast Fourier Transform. 17, 42
FNT	Fermat Number Transform. 17, 42, 48
GHz	Gigahertz. 58, 78
GiB	Gibibyte. 45
ICMNT	Inverse Complex Mersenne Number Transform. 43, 46, 47
INTT	Inverse Number Theoretic Transform. 41
IP	Internet Protocol. 9, 19–21, 29, 30, 50, 55, 57–60, 64, 65, 67, 83
IPv4	Internet Protocol version 4. 9, 11, 18, 19, 21, 50–52, 55–58, 60, 83
IPv6	Internet Protocol version 6. 19, 51
ISP	Internet Service Provider. 7, 28, 49, 52, 53, 55, 64, 68, 69, 84
MDS	Maximum Distance Separable. 8, 15–17, 21, 29, 30, 37, 48, 82
MiB	Mebibyte. 70–72, 74
MSS	Maximum Segment Size. 68, 69
MTU	Maximum Transmission Unit. 51–53, 68, 69
NAT	Network Address Translation. 29, 64
NSA	United States National Security Agency. 3
NTT	Number Theoretic Transform. 16, 17, 37, 39, 41, 42, 48
P2P	Peer-to-Peer. 1–9, 11, 13–16, 19, 21–27, 29, 30, 33, 35–38, 49, 50, 63–71, 74, 76–85
RS	Reed-Solomon. 16, 17, 30, 37, 38, 48, 82
TCP	Transmission Control Protocol. 68, 69, 83
XOR	eXclusive OR. 16, 40

LIST OF MISTRUSTFUL P2P MODEL SYMBOLS

α	Minimum backoff time between consecutive requests. 33, 34
$\bar{\tau}$	Estimated average block transfer time. 33, 34, 69
β	Backoff time scale factor for requests not disclosing block ownership information. 33, 34
δ	The difference between the actual and the configured sizes of the largest colluding attacker. 66
$\epsilon(k)$	Erasure coding overhead. 24, 30
η	Backoff time increase factor for requests implicitly disclosing block ownership information (exponential factor). 33, 34
λ	Backoff time increase factor for requests not disclosing block ownership information (linear factor). 33, 34
μ	Maximum backoff time between consecutive requests. 33, 34, 69
ρ	No. of peers currently participating in the content sharing (active). 34
σ	Minimum no. of blocks disclosed to any of the top c peers $\left[1 \leq \sigma \leq {}^{m}/_{c}\right]$. 66
^{p}	For backoff time related symbols, the p index refers to the peer component of the backoff time (between requests to the same peer). 34, 69
$_{s}$	For backoff time related symbols, the s index refers to the swarm component of the backoff time (between requests to the same swarm). 34, 69
b	Calculated backoff time. It is used to determine the actual randomly generated backoff time $\left[\text{between } 0 \text{ and } b\right]$. 33, 34, 69
c	Size of the largest attacker to be protected against (no. of unique peers) $\left[1 \leq c \leq m\right]$. 24, 25, 29–31, 34–36, 63–67, 69, 70, 74, 75, 78, 81, 82, 84
k'	No. of blocks required to fully download a content $\left\lceil k\left(1+\epsilon(k)\right)\right\rceil$. 24, 30
k	No. of chunks into which a content has been partitioned. 24, 25, 29, 30, 34, 36, 65, 66, 69, 70, 75, 78, 82, 84, 85
m	Minimum network disguise overhead (in blocks), and maximum no. of blocks that can be disclosed to any set of c peers. 24, 25, 27, 29–31, 34–36, 63, 65–67, 69, 70, 74, 75, 78, 81, 82, 84
n	No. of blocks generated. 24, 30
u	No. of consecutive requests not disclosing block ownership information (requests that have been refused). 33, 69
v	No. of consecutive requests implicitly disclosing block ownership information (requests that have been either canceled or interrupted). 33
w_{η}	Multiplicative weight factor to apply when a block offer is canceled. 32, 33
w_{λ}	Additive weight factor to apply when a block offer is accepted. 32, 33
w_{μ}	Weight block decrease to apply when a block offer is accepted. 32

w_i Block i's weight considered for the random weighted selection of a block to offer. 32

w_s Weight of a block that was recently downloaded (starting weight). 32, 33

List of Erasure Code Symbols

$L'(x)$	Derivative of Lagrange basis polynomial, $L(x)$. 44, 46
$L(x)$	Lagrange basis polynomial $\left[\prod_{\alpha=0}^{k-1}(x-x_\alpha)\right]$. 39, 44, 46, 47
$M(k)$	Time complexity of multiplying two polynomials of degree less than k over a finite field. 41, 43, 44, 47, 48
$N(k)$	Time complexity of performing an NTT over a finite field. 41, 43, 47
$S_{\alpha,j}$	The result of evaluating polynomial $s_j(x)$ at point W_k^α $\left[s_j\left(W_k^\alpha\right)\right]$. 41, 42, 46
$W_\eta^{z_l}$	Power representation of point x_l, when it is an η-th unity root. 47
W_k^i	Unity root i within the set of the k-th roots of unity. 41, 42, 46, 47
W_k	A k-th root of unity. 41
$Y_j(x)$	Polynomial of degree $\eta-1$ composed of no more than k non-zero coefficients, $y_{l,j}$, each at the position corresponding to the power representation $W_\eta^{z_l}$ of the interpolation point x_l $\left[\sum_{\alpha=0}^{k-1} y_{\alpha,j}\cdot x^{z_\alpha}\right]$. 47
$\epsilon(k)$	Erasure coding overhead. 15, 16, 37
η	Cardinality of the smallest set of unity roots that contains all k interpolation points. 46–48
$\mathbb{F}_p$	A prime field comprising the finite set $\{0,\ldots,p-1\}$ of p elements. 15, 17, 39
$\mathbb{F}_q$	A finite field with q elements. It is either a prime field, $\mathbb{F}_p$, or an extension field, $\mathbb{F}_{p^m}$. 38, 40, 41
$\mathbb{F}_{2^m-1}$	Mersenne finite field. A prime field whose cardinality is a Mersenne prime. 40, 41, 45
$\mathbb{F}_{2^m}$	Binary finite field. It is either the prime field $\mathbb{F}_2$ or an extension field. 16–18, 39
$\mathbb{F}_{2^{2^m}+1}$	Fermat finite field. A prime field whose cardinality is a Fermat prime. 18, 47, 48
$\mathbb{F}_{(2^m-1)^2}$	Complex Mersenne finite field. It is an extension field akin to complex numbers $\left[\{a+b\hat{\imath}\mid a,b\in\mathbb{F}_{2^m-1}\}\right]$. 8, 30, 37, 39–41, 45, 47, 48, 83
$\mathbb{F}_{p^m}$	An extension over the prime field $\mathbb{F}_p$ whose elements are polynomials of degree $m-1$ with coefficients from $\mathbb{F}_p$. 15, 39
$\mathbf{B}$	Matrix representation of an encoded content ($n\times d$). Rows represent blocks, b_l, and columns represent encoded vectors, e_j. 38, 39, 45
$\mathbf{C}$	Matrix representation of a content ($k\times d$). Rows represent chunks, c_i, and columns represent source vectors, s_j. 38, 39, 45
$\mathcal{F}$	Transformation $\mathbb{F}_q^k\rightarrow\mathbb{F}_q^n$ that generates an encoded vector e_j from a source vector s_j as $(s_{0,j},\ldots,s_{k-1,j})\xrightarrow{\mathcal{F}}(e_{0,j},\ldots,e_{n-1,j})$. 38, 45, 46

$\mathcal{T}$ — Transformation that defines the encoding process $\left(\mathbf{C} \xrightarrow{\mathcal{T}} \mathbf{B}\right)$. Matrix $\mathbf{B}$ is generated by applying the transformation $\mathcal{F}$ to each one of the d columns of matrix $\mathbf{C}$. 38, 39, 45

ω_i — Barycentric weight of Lagrange interpolation method for point x_i $\left[L'(x_i) = \left(\prod_{j=0, j\neq i}^{k-1} (x_i - x_j)\right)^{-1}\right]$. 39, 44, 46, 47

ρ — Radix of the NTT. 42

σ — No. of recursive stages used by multi-point polynomial evaluation and polynomial interpolation algorithms $\left[\log_2 k\right]$. 43, 44

b_l — The set of erasure coded symbols composing block l $\left[(s_0(x_l), \ldots, s_{d-1}(x_l)) = (e_{l,0}, \ldots, e_{l,d-1})\right]$. 38

c_i — The set of source symbols composing chunk i $\left[(s_{i,0}, \ldots, s_{i,d-1})\right]$. 38

d — No. of symbols composing a chunk or a block. 38, 39, 45

e_j — Encoded vector of size n composed of the erasure coded symbols at position j of each block $\left[(s_j(x_0), \ldots, s_j(x_{n-1})) = (e_{0,j}, \ldots, e_{n-1,j})\right]$. 38

$e_{l,j}$ — Erasure encoded symbol from block l at position j $\left[s_j(x_l)\right]$. 38, 39, 43, 45–47

k' — No. of polynomial coefficients at stage u of multi-point polynomial evaluation and polynomial interpolation algorithms $\left[k/2^u\right]$. 44

k — No. of data symbols. 8, 15–18, 30, 37–39, 41–48, 82, 83, 85

n — No. of erasure coded symbols. 8, 15–18, 30, 37–41, 45–48, 82, 83

$p_{u,v}$ — v-th polynomial used by multi-point polynomial evaluation and polynomial interpolation algorithms at stage u $\left[\prod_{\alpha=0}^{k'-1} (x - x_{v \cdot k' + \alpha})\right]$. 43, 44

p — A prime number. 8, 15, 17, 30, 37, 39, 40, 45

q — Cardinality of a finite field (either a prime field or an extension field). 15, 16, 39, 40

r_0 — Left-side stage result of multi-point polynomial evaluation or polynomial interpolation algorithms. 43, 44

r_1 — Right-side stage result of multi-point polynomial evaluation or polynomial interpolation algorithms. 43, 44

r — Primitive root of a finite field $\mathbb{F}_q$ $\left[\mathbb{F}_q = \{0, r^0, r^1, \ldots, r^{q-2}\}\right]$. 40, 41

$s_j(x)$ — Polynomial representation of the source vector s_j $\left[\sum_{\alpha=0}^{k-1} s_{\alpha,j} \cdot x^{\alpha}\right]$. 38, 39, 41, 43, 44, 47

s_j — Source vector of size k composed of the source symbols at position j of each chunk $\left[(s_{0,j}, \ldots, s_{k-1,j})\right]$. 38, 45

$s_{i,j}$ — Source symbol from chunk i at position j. 38, 39, 41–43, 45, 46

u — Multi-point polynomial evaluation or polynomial interpolation stage. 43, 44

v — Index of a polynomial used by multi-point polynomial evaluation and polynomial interpolation algorithms at a stage u. 44

x_i — Point or code locator i. 39, 43, 44, 46, 47

$y_{l,j}$ — The product of $e_{l,j}$ by ω_l $\left[e_{l,j} \cdot \omega_l\right]$. 39, 44, 46, 47

LIST OF NS-3 SIMULATION SYMBOLS

$\bar{n}$	Average no. of routing table entries at nodes within the path between source and destination. 20, 21
h	No. of hops between source and destination nodes. 19–21
n_i	No. of routing table entries at node i. 20, 21
n	No. of routing table entries. 19–22

1 INTRODUCTION

> *Historically, privacy was almost implicit, because it was hard to find and gather information. But in the digital world, whether it's digital cameras or satellites or just what you click on, we need to have more explicit rules - not just for governments but for private companies.*
>
> **— Bill Gates**

In the last two decades, the advances in computer technology have drastically changed the media landscape. The widespread availability of digital media and broadband Internet connections enabled consumers to become also producers and distributors of creative work, something that in the past was mostly limited to professional parties. Peer-to-Peer (P2P) networks endowed individuals with the means to easily and efficiently distribute digital media over the Internet, and are extensively used for large-scale file sharing due to their decentralized and scalable nature. The P2P architecture enables solutions that enhance privacy, such as Freenet [@6], Nymble [63], and Tor [18], but user legal liability issues may be raised as it also facilitates unauthorized distribution and reproduction of copyrighted material [58].

The reasons behind seeking privacy may be various, such as to avoid user profiling, tracking and data mining, to privately download legal contents that may be embarrassing or objectionable from a political, religious or social point-of-view, or to download ethical contents that are considered illegal or incriminating without being subject to any liability. The current state of computer technology enables organizations and individuals to create databases of personal information that were previously impossible to set up, and swap this information, sell it or use it in any other way as a commodity [66]: organizations and individuals are able to collect information about users, combine facts from separate sources, and merge them to create such databases of personal information. Users may feel exposed or embarrassed if it becomes public that they had access to contents such as to help dealing with alcohol and drug abuse, to help dealing with anger management, or considered heretical or profane. Even accessing illegal or incriminating contents may have an ethical motivation because laws are, ideally, drawn from ethics, but that is not always the case: e.g., autocratic regimes consider illegal to share or even access contents that they consider inappropriate or potentially harmful.

Traditional P2P file sharing systems focus on performance and scalability, disregarding any privacy issues that may arise from their use. They take advantage

of the large number of interconnected peers[1], and their idle resources, to more efficiently distribute contents at the cost of requiring peers to publicly advertise what they download, making it trivial to identify user content interests. This problem is further aggravated by the fact that peers form interest-based communities, and every single connection presents an opportunity for a malicious peer to passively obtain additional information that may enable the identification of user content interests, with high certainty, by monitoring just a small fraction of the network [11].

Several privacy-preserving P2P systems have been proposed to address the user profiling and content interest identification issues, such as BitBlender [2] and SwarmScreen [11], but they still require peers to advertise, either fully or partially, what they download. Moreover, they do not address any legal issues that may arise from their legit and well-intended use. P2P networks facilitate unauthorized distribution and reproduction of copyrighted material [58], and are usually connoted with copyright infringement because it is estimated that, over the course of a month, 96.3% of users of BitTorrent portals have downloaded at least one infringing content [47]. Therefore, P2P file sharing users aiming at protecting their privacy may also be held legally liable for, unknowingly and unwillingly, downloading an illegal content due to a misleading description or as a result of using an insidious resource to locate it. Lacking alternatives, users have adopted general-purpose anonymity systems for P2P file sharing [8], misunderstanding the privacy guarantees provided by such systems [9], in particular when relaying traffic of insecure applications [35]. Anonymity systems provide a channel to anonymously transmit messages, but the user's identity may be disclosed by the content of that messages, which are the sole responsibility of the application being used.

P2P file sharing encompasses both public (open access) and private (restricted access) file sharing. The access control mechanisms may be used to prevent misbehaving peers from joining the network or to ban them (network level), and to constrain the distribution of a given content (content level). Lacking any access control mechanisms, public P2P file sharing presents additional privacy and legal challenges because no trust between peers can be assumed, malicious and colluding peers cannot be easily banned from the network, all contents are public, easing the identification of user content interests, and users are free to publish contents without being subject to any legal or ethical assessment or any content description (metadata) truthfulness validation. Thereby, this work considers public P2P file sharing in large groups of untrusted peers (untrusted P2P networks) to support the most privacy demanding file sharing scenario.

[1] The term *peer* is used to refer to the network node, and the term *user* is used to refer to the person.

1.1 Legal and Ethical Framework

Computer ethics is a field of study that addresses the ethical challenges of computer technology but several lines of thought exist. Therefore, the aim of this section is to describe the ethical challenges considered to be more relevant to this work. The reader is referred to [29] and [32] for a wide and thorough explanation of these challenges. For the legal and privacy dimensions of P2P file sharing, based on the intellectual property law, in particular on the copyright law, the key legal aspects to take into consideration on Western countries regarding direct and indirect user liability are highlighted.

Laws are formally adopted rules that mandate or prohibit a certain behavior, created by the members of a society to balance the individual rights to self-determination against the needs of the society as a whole, and, ideally, are drawn from ethics, which define what is considered right or wrong, i.e., the socially acceptable behaviors. The key difference between laws and ethics is that the former carry the authority of a governing body, usually a nation [66]. In turn, ethics are based on cultural mores, beliefs, values and principles, which reflect the unique existential experiences that are accumulated as individuals as well as societies and, supported on institutions, provide long-term stable rules that are made obvious. Thus, these rules can be seen as refractions of the common world awareness that give rise to different experiences and interpretations: multicultural ethics [6].

The cultural differences, despite some ethical standards being universal (e.g., murder and theft), make it difficult to define what is ethical or not. Studies have shown that the perspective on ethical practices of individuals regarding the use of computer technology differs with their nationality. Asian traditions of collective ownership conflict with Western protection of intellectual property, and many of the ways the former use software is considered software piracy by the latter [66]. These differing perspectives are also evident on the control over the Internet content and on the surveillance made by governments: they radically differ between countries [65]. According to a report [4] from the Reporters Without Borders, the *Great Firewall* of China is getting "taller", the United Kingdom is the "world champion of surveillance", and "NSA[2] symbolizes intelligence services' abuses"; on the other hand, Norway, Sweden, and Finland are at the top of the World Press Freedom Index 2017 [@2].

The utmost objective of intellectual property law is to promote progress by encouraging and stimulating human intellectual creativity and broad dissemination of its result [26]. Creators are granted exclusive rights as an incentive to continue their works, and, at the same time, the society can benefit from their broad dissemination. Copyright is a legal device that protects the expression of an idea by conferring the creator, for a period of time, exclusive rights to publish, sell and control the reproduction of his work, whom may grant or sell these rights to others. Copyright infringement and piracy are then violations of one of these exclusive rights [13]. Still,

[2] United States National Security Agency (NSA).

these rights are granted nationwide, not worldwide: the Berne Convention [@1] is not ratified by all countries and only sets the minimum standards for copyright. Thus, given that P2P users are spread all over the world and a single content sharing may cross many jurisdictions, it is hard to determine the applicable legal framework and the extent of user liability. Even when the legal framework can be clearly defined, it may still be difficult to determine the extent of user liability since the copyright rights may conflict with other interests and rights, e.g. privacy rights, being subject to proportionality assessment to balance all the rights and interests at stake [30].

The enforcement of copyright rights on the private sphere of users may clash with their privacy rights, thereby, when present, the private copying law introduces an exception to these exclusive rights under some conditions. For private use and for ends that are neither directly nor indirectly commercial, a natural person[3] is allowed to copy copyrighted works on the condition that the rightholders receive fair compensation. Therefore, any media that may be used for the reproduction of copyrighted works by consumers are designated for payment of the private copying levy, which will compensate rightholders for any harm that may be caused [45, 48]. The private copying law, when present, differs considerably between countries [31], reflecting the lack of consensus and the complexity of this subject. Moreover, it is not clear that private copying always causes harm – e.g., it may increase the group of fans and enables users to try before buying –, and not all media are used for the reproduction of copyrighted works. The reader is referred to [64], [26], and [48] for further discussion on the subject.

P2P networks connect users all over the world without considering either international borders or culture. It is not feasible to enforce the exclusive rights provided by copyright law, if present, in every single jurisdiction while ensuring that they do not conflict with the rights and interests of users. As so, in an attempt to mitigate the impact of copyright infringement, rightholders are now trying to block websites such as The Pirate Bay instead of trying to bring end users to court [53] because, in general, it is more efficient and less onerous to prove that the owners of such websites profit from the copyright infringement. Therefore, such websites are subject to indirect liability as they induce, contribute to or fail to prevent direct copyright infringement. Nevertheless, this does not mean that sharing copyright infringing contents is legal. On the contrary, users may be subject to civil and potentially criminal liability [48]. The extent of such liability depends on the applicable legal framework and on the intent of such acts: a user may also be subject to indirect liability.

The multitude and complexity of copyright laws across the globe make it impossible to define clear boundaries regarding user liability. Still, the users of P2P systems are not expected to be held legally liable for, unknowingly and unwillingly, downloading an illegal content (direct liability) or contribute to its download (indirect

[3] In jurisprudence, a natural person is a human being, as opposed to a legally generated juridical person.

liability) if the main motivation to use the P2P system is to share legit contents, and it cannot be proven that the user had access, either in part or fully, to the content data. The user does not benefit directly from the download of an illegal content that he has no access to, and he is also unable to determine that the content is unexpectedly illegal before having access to its data. Therefore, it is plausible that he either has been mislead into downloading it or had no intention in assisting a misbehaving peer infringing copyright law.

In sum, the basic ethical imperatives are that a person should not, knowingly and willingly, cooperate in or contribute to the wrongdoing of another, and that the human intellectual creativity needs to be encouraged and stimulated in order to promote progress. The extent of these incentives depends on the cultural background as the well-being of the society may be incentive enough (collective ownership) or further incentives may be required (intellectual property rights). P2P networks introduce new challenges to the private copying levy system, and the legislation on this subject is expected to change in the near future to address them. Users of P2P systems should not be subject to any civil or criminal liability as a result of legitimate usage of the system if the main motivation to use it is to share legit contents, and they have no access to the content data. The intent is important to determine the extent of user liability, especially if his actions directly or indirectly caused provable harm.

1.2 Traditional Peer-to-Peer Systems

Traditional P2P file sharing systems disregard user's privacy as their focus is on performance and scalability. BitTorrent is the most prominent of such systems, therefore it is used as a representative example. First, it is provided a brief definition of common BitTorrent terminology – *hash*, *content*, *chunk*, *torrent file*, *peer*, *seeder*, *leecher*, *swarm*, and *tracker*. Then, its content sharing process is described. Lastly, the main privacy and legal issues are presented.

A *hash* is an alphanumeric string used to identify and to verify the integrity of the data being transferred, usually an SHA-1 digest (hexadecimal string). A *content* is composed of one or more files, identified by the hash of its data, and partitioned into several pieces. A *chunk* is one of those data pieces, and a *torrent file* provides the content's metadata. A *peer* is a node sharing chunks, which, for a given content, can be either a *seeder* – a peer that has all data chunks and is just sharing them – or a *leecher* – a peer still downloading the content and sharing the chunks it has already downloaded. Table 1.1 depicts the peer roles and their sharing behavior. A *swarm* is the set of peers sharing the same content (peers form interest-based groups to share contents), and is usually identified through the hash of the content. A *tracker* is a central node that provides lists of peers in swarms by keeping track of which peers are sharing which contents and their role (seeder or leecher) on each content.

Table 1.1: *Peer roles and sharing behavior on traditional P2P systems.* A seeder has all data chunks and just shares them; a leecher is a peer still downloading missing data chunks and sharing the data chunks it has already downloaded.

Peer Role	Chunks		
	Owned	To Share	To Download
Seeder	All chunks	All chunks	None
Leecher	Downloaded chunks	Downloaded chunks	Missing chunks

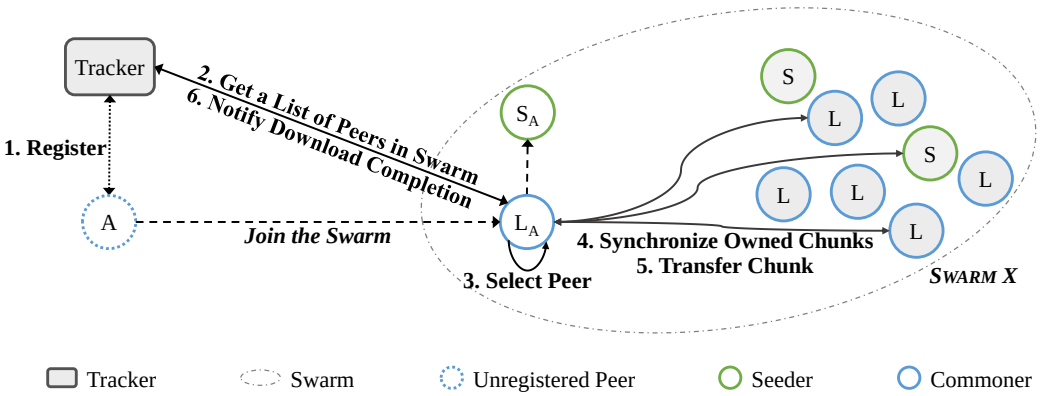


Figure 1.1: *Overview of the content download process on traditional P2P systems.* First, peer A registers at the tracker to join the swarm of content X (swarm X). After registering, the peer starts as a leecher (still downloading) and requests from the tracker a list of peers (a subset) sharing that content. Until download completion, peer A selects peers from that list, synchronizes the chunks it owns with theirs, and transfers the missing chunks; the list of peers may be updated during content download. Upon download completion, peer A notifies the tracker to be known as a seeder.

The main performance metric considered by traditional P2P file sharing systems is the average download time or the average download bitrate, which are two sides of the same coin. BitTorrent protocol employs several mechanisms for chunk and peer selection, such as rarest first mechanism – locally rarest chunks are downloaded first –, and optimistic unchoking – periodically select a random peer – aiming at constantly improving the download bitrate [14]. The distinction between seeders and leechers is used to estimate relative download times (through their ratio), and also to quickly assess the content availability[4], which is assured if at least a single seeder is present. The steps required to download a given content, depicted in Figure 1.1, can be summarized as follows: 1) a peer willing to download a given content registers at the tracker and joins the swarm; 2) it requests a list of peers (a subset) in the swarm from the tracker; 3) it selects peers from that list to obtain missing chunks in order to complete its download; 4) peers synchronize the chunks they own and miss to determine if any missing chunks can be transferred; 5) if selected peers own missing chunks, those chunks get transferred; 6) upon download completion, the peer notifies the tracker to be known as a seeder for that content. The list of peers may be updated during content download.

[4] A content is considered to be available if it can be fully downloaded.

The main privacy issues of traditional P2P systems are that they publicly disclose user content interests, and provide a proof that the user is able to access a content in part or entirely; the former discloses an intention while the latter provides a proof of its realization. A peer only downloads the contents that a user is interested in, therefore, by registering at a tracker and joining swarms, the user content interests are being publicly disclosed. Peers provide a proof that the user is able to access a content in part or entirely by advertising which chunks they own, and entirely by notifying the tracker upon download completion.

The main legal issue of traditional P2P systems is that, unknowingly and unwillingly, a user may be held liable for copyright infringement due to illegal content download. If a content is published with a misleading description or if the resource used to obtain the hash of the content is insidious, the user will only become aware of that fact after accessing the content data in part or fully. For contents that can be accessed in part before fully downloading them, which is usually the case of multimedia contents, the user may be infringing copyright law after downloading a single or a few chunks. The copyright infringement can be trivially proved because peers advertise which chunks they own and notify the tracker upon download completion.

1.3 Problem Definition

In the context of this work, privacy preservation is defined as the concealment of user content interests. Therefore, the aim is at developing a privacy-preserving P2P file sharing model that:

Hides user content interests. The user must be able to hide his content interests from any participant in the system: trackers, regular peers, or groups of colluding peers. Protection against external entities monitoring all the user's traffic, such as Internet Service Providers (ISPs) or governments, is out of the scope of this work.

Has no trust requirements. The user must be able to download a content without having to trust anyone. Therefore, the privacy-preserving P2P model must enable content sharing in large groups of untrusted peers (untrusted P2P networks).

Prevents user liability. The user shall not be subject to any liability as a result of legitimate usage of the privacy-preserving P2P model. The user shall be protected against actions of misbehaving peers, or from the download of contents published with misleading description (misleading contents), which may drive users to unknowingly and unwillingly download unethical or illegal contents.

Enables timely downloads. The user should be able to download a content in due time. The average download bitrate must be within the same order of magnitude of traditional P2P systems that do not preserve the privacy of users.

1.4 Goals and Contributions

The main goal of this thesis is to prove the following statement, also assumed as the working hypothesis: *It is possible to deterministically hide the content interests of users sharing publicly accessible files over untrusted P2P networks without disclosing what peers download or miss.*

This work has three main contributions. The first and core contribution is the proposal of a novel P2P file sharing model, named Mistrustful P2P as it is built on the concept of mistrusting all the entities participating in the P2P network. This model resorts on erasure codes to avoid disclosing what peers download or miss, and its validation was conducted in ns-3 network simulator [@4]. The second main contribution is the proposal of Storm erasures codes. Storm erasure codes are more suited for P2P file sharing because they do not introduce network overhead and have efficient encoding and decoding algorithms: the network is typically the most constrained P2P file sharing resource, not the Central Processing Unit (CPU) [36]. The third main contribution is the proposal of CIDRarchy ns-3 routing protocol[5]. The Mistrustful P2P model cannot be validated analytically or using real large-scale implementations, and small scale simulations, although feasible, may not be enough as some issues may only arise at the scale of thousands of peers or more [10]. CIDRarchy improves the time complexity of packet forwarding in ns-3 for hierarchical networks, such as the Internet, because existing routing algorithms preclude the simulation of content sharing with thousands of peers in due time. The following paragraphs describe each main contribution in more detail.

Mistrustful P2P Model. It is a novel P2P file sharing model that provides plausible deniability through *content interest disguise*: hides user content interests by downloading both contents that the user is interested in (genuine) and additional contents of no interest to the user (cover), as long as they cannot be distinguished. This model has no trust requirements, prevents user legal liability in case of legitimate usage, and ensures deterministic protection of user content interests against attacks of a size up to a configured level. Therefore, users are not required to establish trust links in order to participate in the content sharing. It also enables each user to set the required trade-off between privacy and performance by configuring, per content, the size of the largest group of colluding peers to be protected against, and the minimum network overhead of content interest disguise.

Storm Erasure Codes. These erasure codes are a rateless Maximum Distance Separable (MDS) construction of Reed-Solomon codes [50] over the finite field $\mathbb{F}_{p^2}$, where p is a Mersenne prime. To the best of the author's knowledge, this is the first rateless construction (n can be increased in steps of k) with $\Theta(n \log k)$ encoding time complexity and $\min\left\{\Theta(n \log n), \Theta\left(k \log^2 k\right)\right\}$ upper bound for decoding time complexity. Storm aims at demonstrating that erasure codes can be used to enable

[5] The meaning of routing protocol in the context of the ns-3 network simulator differs from the canonical one. The differences are described in Section 2.3.

Mistrustful P2P in a large set of devices and without depending on any patented erasure code.

CIDRarchy ns-3 Routing Protocol. CIDRarchy is an ns-3 IPv4[6] routing protocol developed for ns-3 that uses Classless Internet-Domain Routing (CIDR) as the base to create a hierarchical Internet-like network topology that enables IP packet forwarding with constant time complexity and automatic IPv4 address assignment, and the implementation of an ns-3 helper to ease network topology creation. CIDRarchy enables the validation of the Mistrustful P2P model for contents with thousands of peers because the time gains over built-in ns-3 routing protocol implementations can reach over one order of magnitude.

Three conference papers and one journal paper were published as a direct result of this thesis.

[1] Silva, P. M. da, Dias, J., and Ricardo, M. "CIDRarchy: CIDR-based ns-3 Routing Protocol for Large Scale Network Simulation". In: *Proceedings of the 8^{th} International Conference on Simulation Tools and Techniques*. SIMUTools '15. Athens, Greece, 2015, pp. 267–272.

[2] Silva, P. M. da, Dias, J., and Ricardo, M. "Storm: Rateless MDS Erasure Codes". In: *Wireless Internet: 8^{th} International Conference, WICON 2014, Lisbon, Portugal, November 13-14, 2014, Revised Selected Papers*. Ed. by Mumtaz, S., Rodriguez, J., Katz, M., Wang, C., and Nascimento, A. Springer International Publishing, 2015, pp. 153–158.

[3] Silva, P. M. da, Dias, J., and Ricardo, M. "Mistrustful P2P: Privacy-preserving File Sharing Over Untrustworthy Peer-to-Peer Networks". In: *Proceedings of IFIP Networking 2016*. IFIP Networking '16. Vienna, Austria, 2016, pp. 395–403.

[4] Silva, P. M. da, Dias, J., and Ricardo, M. "Mistrustful P2P: Deterministic Privacy-preserving P2P File Sharing Model to Hide User Content Interests in Untrusted Peer-to-Peer Networks". In: *Computer Networks* 120 (2017), pp. 87–104.

1.5 Thesis Structure

The remainder of this thesis is structured as follows. Chapter 2 describes the related work for each one of the three main contributions. Sections 2.1, 2.2, and 2.3, respectively, the related work for privacy-preserving P2P file sharing systems, erasure codes, and existing ns-3 routing protocols. Chapter 3 presents the Mistrustful P2P model. Storm erasure codes and CIDRarchy ns-3 routing protocol are presented and evaluated, respectively, in Chapters 4 and 5. The Mistrustful P2P model is validated in Chapter 6, and the thesis conclusions are drawn in Chapter 7.

[6] Internet Protocol version 4.

Related Work

2

> *If privacy is outlawed, only outlaws will have privacy.*
>
> — **Philip Zimmermann**

This chapter presents the related work for privacy-preserving P2P systems, erasure codes, and ns-3 routing protocols. Section 2.1 depicts privacy-preserving P2P systems designed specifically for P2P file sharing and providing privacy through plausible deniability, given that they typically introduce less overhead and may take advantage of it to improve the overall performance. Section 2.2 presents erasure codes more suitable for P2P file sharing that introduce no network overhead because the network is typically the most constrained P2P file sharing resource, not the CPU [36]. Section 2.3 describes existing ns-3 IPv4 routing protocols for infra-structured networks, such as the Internet, that were readily available as of version 3.22 of ns-3 network simulator. For each section, it is provided the required background before presenting the related work, and a summary after its presentation.

2.1 Privacy-preserving Peer-to-Peer Systems

Section 2.1.1 describes the two main classes of privacy-preserving P2P systems, and the main techniques they employ. Sections 2.1.2 through 2.1.5 present each one of the related work systems. These systems are compared in Section 2.1.6.

The description provided for each system focus on the trust requirements, on the protection against both passive and active attacks aiming at identifying user content interests, and on the potential legal liability of users while using the system for legitimate purposes.

2.1.1 Background

Several privacy-preserving P2P systems have been proposed, but, given that there is a trade-off between privacy and performance, they consider different attack models and employ different techniques for privacy preservation. The majority of such systems provides privacy through anonymity, which is usually stronger but has lower performance, or through plausible deniability, which is usually weaker but has better performance.

Privacy Through Anonymity

Anonymity systems provide privacy preservation by concealing the identity of the peer, but not necessarily its actions: an attacker may be able to create a user profile, but it should not be able to link a peer to that profile. Peers communicate through indirect paths so that each intermediary peer knows only the location of the immediately preceding and following peers, thus hiding the identity of the communication endpoints. These systems employ techniques such as *onion routing* [23] and *information slicing* [33] to provide anonymous communication. *Onion routing* is a technique that encapsulates messages in layers of encryption, which are removed one by one at each hop to determine the next hop towards destination, so that only the intended recipient is able to decode the message while hiding the identity of the sender. *Information slicing* is presented by its authors as an alternative to *onion routing* that resorts on sending slices of information through disjoint paths so that only the intended recipient is able to receive all slices and, by doing so, decode the message. It enables to exchange a symmetric key without resorting on public key cryptography, and does not require multiple encryption layers, as long as there are at least two disjoint paths.

Anonymity systems introduce computational and network traffic overhead to hide the identity of the communication endpoints, but do so without improving the overall performance of the network. General-purpose anonymity systems, such as Tor [18], provide a channel to anonymously transmit messages but cannot prevent the disclosure of user's identity when relaying traffic of insecure applications, such as BitTorrent, because the content of that messages is the sole responsibility of the application. The relay peers are also more prone to be held legally liable, in particular the last peer on the path because it appears as the source and has access to the message content. E.g., Tor defaults to a path length of four (300% network overhead), which lowers the throughput and increases the average latency [2], and a Tor relay node (core router) may relay traffic of misbehaving peers, which makes the last one on the path (exit node) to appear to be the originator of such traffic.

Privacy Through Plausible Deniability

Privacy through plausible deniability consists in introducing uncertainty so that an attacker fails to ascertain any sensitive information, and therefore a user can plausibly deny any claim of such attacker. It is achieved by employing techniques such as *request relaying* and *content interest disguise*. *Request relaying* creates uncertainty about the communication endpoints by introducing relay peers in order to thwart any attacks aiming at identifying the role of a peer on a chunk transfer: requester, relay or provider. *Content interest disguise* hides user content interests by enabling peers to communicate directly but requiring them to download additional contents. Thus, content download no longer means interest in a given content, and user content interests can longer be identified by monitoring just a small fraction of the network [11]. *Request relaying* and *content interest disguise* may be used together.

Privacy-preserving systems providing privacy through plausible deniability introduce network traffic overhead to create uncertainty, but that overhead may be used to improve the overall performance of the network: e.g., relay peers may cache contents and therefore increase their availability. The privacy protection provided by these systems may not be enough for some users, given that an attacker may be able to determine the set of all contents potentially downloaded.

2.1.2 BitBlender

BitBlender [2] provides plausible deniability by introducing relay peers that simply proxy requests on behalf of other peers. Peers willing to act as relay peers can register at a central node called *blender*, and, once requested, will join a P2P swarm in a probabilistic way so that they cannot be distinguished from regular peers. The joining probability of relay peers is defined by the *blender*, when asking registered peers to join a P2P swarm, so that the set of relay peers remains unknown while having the cardinality requested by the tracker. It enables the identification of regular and relay peers through download progress tracking, and provides a trivial proof of content download because peers advertise what they download. Thereby, BitBlender provides protection against passive attacks, but it is vulnerable to active attacks using download progress tracking. It requires users to trust both the tracker and the *blender*. Its users may be subject to legal liability for downloading misleading contents, and for relaying traffic of misbehaving peers.

2.1.3 SwarmScreen

SwarmScreen [11] provides plausible deniability by obscuring user content interests through content interest disguise. The devised scheme, which consists in "adding a small percentage (between 25% and 50%) of additional random connections that are statistically indistinguishable from natural ones", thwarts guilt-by-association attacks, that is, attacks in which the user content interests can be inferred with high certainty just by classifying peers based on the behavior of the communities they participate in. SwarmScreen has no trust requirements, but its attack model only considers passive attacks. It is vulnerable to active attacks because contents can be distinguished through download progress tracking. Peers communicate through direct links, but users may be subject to legal liability due to the download of misleading contents.

2.1.4 The BitTorrent Anonymity Marketplace

The BitTorrent Anonymity Marketplace [42] follows SwarmScreen's approach to provide plausible deniability. It does not present any trust requirements, and peers also communicate through direct links. However, in order to protect against both passive and active attacks, all contents are fully downloaded to make them indistinguishable, given that an attacker is able to fully track download progress: peers advertise what they own and miss. The authors define *k-anonymity* as the privacy protection level

Table 2.1: *Privacy-preserving P2P file sharing systems comparison.* The comparison considers trust requirements, protection against passive and active attacks aiming at identifying user content interests, and potential user legal liability.

Work	Has No Trust Requirements	Protects Against Attacks		Prevents Legal Liability
		Passive	Active	
BitBlender	✗	✓	✗	✗
SwarmScreen	✓	✓	✗	✗
The BitTorrent Anon. Mark.	✓	✓	✓	✗
Petrocco et al.	✗	✓	✓	✗

obtained from fully downloading k contents. Thus, since it introduces high network overhead, it either prevents downloads from timely completing or constrains the achievable level of privacy protection. Users may be subject to legal liability due to the download of misleading contents.

2.1.5 Petrocco et al.'s

Petrocco et al. [44], also following SwarmScreen's approach, proposed a system that aims at hiding user content interests without compromising timely download completion. Their system relies on private swarms, *request relaying*, caching, and partial advertisement of downloaded chunks. As stated by the authors, private swarms are required to ensure a good level of privacy, i.e., to avoid the identification of user content interests through download progress tracking. Yet, to obtain the credentials needed to join a private swarm, peers must trust one or more participants. Also, as only a fraction of the chunks are advertised, it is not clear how a content sharing is bootstrapped with few seeders or how request relay should operate during periods of content unavailability. This system provides protection against passive and active attacks, but its users may be held legally liable due to relay traffic.

2.1.6 Summary

Table 2.1 summarizes and compares the privacy-preserving P2P file sharing systems previously described taking into consideration the trust requirements, the privacy protection against passive and active attacks, and the potential legal liability of users. BitBlender and SwarmScreen are unable to hide user content interests from active attacks using download progress tracking. The BitTorrent Anonymity Marketplace and Petrocco et al.'s systems are able to thwart both passive and active attacks. BitTorrent Anonymity Marketplace requires peers to fully download all contents in order to make them indistinguishable, and therefore introduces a considerable network overhead that will either prevent downloads from timely completing or constrain the achievable level of privacy protection. Petrocco et al.'s system requires peers to trust one or more participants in order to reduce network overhead by using partial advertisement of downloaded chunks.

All solutions require peers to advertise, either fully or partially, what they have downloaded. An attacker may exploit download progress tracking to obtain a proof that the user is able to access a content either entirely or in part, and to narrow or even void plausible deniability. For unethical or illegal contents that can be accessed in part before fully downloading them, which is usually the case of multimedia contents, an attacker may obtain a proof of such access. As so, the user may be held legally liable for, unknowingly and unwillingly, downloading a single or a few chunks, be it due to traffic relaying or due to misleading content description. Thereby, a legitimate and well-intended usage of these systems may held the user liable for copyright infringement.

2.2 Erasure Codes

Section 2.2.1 provides a brief background of erasure codes, and the concepts introduced in it are described in more detail in Chapter 4 (Storm Erasure Codes). Sections 2.2.2 through 2.2.5 present the related work. The erasure codes are compared in Section 2.2.6.

The description provided for each erasure code focus on two main categories: performance metrics; ability to cope with P2P file sharing dynamics.

2.2.1 Background

An erasure code generates a set of n symbols from a set of k symbols, $k < n$, so that any subset of $k\,(1 + \epsilon(k))$ is enough to reconstruct the original information, where $\epsilon(k)$ is the erasure coding overhead. Erasure codes are usually classified according to three orthogonal properties: systematicity, rate fixedness, and coding overhead. An erasure code is systematic if the input symbols are embedded into the output symbols, and non-systematic otherwise. If n is static and needs to be known before encoding, the erasure code is fixed-rate. If n can be dynamically increased and the amount of symbols that can be generated does not impose any practical limitation, the erasure code is rateless. Finally, an erasure code is MDS if any k symbols out of n are enough to reconstruct the original information $[\epsilon(k) = 0]$, or non-MDS if additional symbols are required $[\epsilon(k) > 0]$. Non-MDS erasure codes reduce the encoding and decoding time complexity orders by introducing coding overhead. The practical encoding and decoding time complexities of any erasure code depend on the efficiency of the integer arithmetic operations they rely on, which are defined over finite fields so that neither rounding nor precision errors are introduced.

A prime field $\mathbb{F}_{q=p}$ is a finite field comprising the finite set $\{0, \ldots, q-1\}$ of q elements, and exists for any prime p. An extension field with $q = p^m$ elements, $\mathbb{F}_{p^m}$, is an extension over the prime field $\mathbb{F}_p$ whose elements are polynomials of degree $m-1$ with coefficients from $\mathbb{F}_p$. Finite fields where $p = 2$ are named binary finite fields. Unlike real arithmetic, finite field arithmetic involves only integer arithmetic,

and the result of an operation over a finite field is also an element of that field. Thereby, it introduces neither rounding nor precision errors. Binary finite fields, since their cardinality is a power of two ($q = 2^m$), are more attractive as they are efficiently representable on computer memory, and the addition over $\mathbb{F}_{2^m}$ is the eXclusive OR (XOR). Multiplication is the logical AND for $\mathbb{F}_2$, but for $m > 1$ it is not as efficient, and it is typically performed via lookup tables for $2 \leq m \leq 16$. These fields support only additive Number Theoretic Transform (NTT), Discrete Fourier Transforms (DFTs) over finite fields, with $\Theta(n \log n \log \log n)$ time complexity [22]. Non-binary finite fields, since their cardinality is not a power of two, are not as efficiently represented on computer memory as binary finite fields, but they support multiplicative NTT, which has $\Theta(n \log n)$ time complexity [5]. Fermat finite fields and Mersenne finite fields – finites fields with cardinality that is, respectively, a Fermat prime ($q = 2^{2^m} + 1$), and a Mersenne prime ($q = 2^m - 1$) –, are just one element away from a power of two. Using up to 128 bits to represent each element ($q \leq 2^{128}$), a Fermat number is prime for $m \in \{0, 1, 2, 3, 4\}$, being 65537 the largest known Fermat prime, and a Mersenne number is prime for $m \in \{2, 3, 5, 7, 13, 17, 19, 31, 61, 89, 107, 127\}$, being $2^{74207281} - 1$ the largest known Mersenne prime.

MDS erasure codes are more suitable for P2P file sharing because they introduce no network overhead, given that the network is typically the most constrained P2P file sharing resource, not the CPU [36]. For other scenarios, non-MDS erasure codes may be more suitable, but they are out of the scope of this work. LT codes [39] and Raptor codes [56] are the most prominent examples of non-MDS erasure codes because they are rateless and asymptotically optimal [$\epsilon(k) \to 0$ as $k \to \infty$], and the latter is able to achieve linear encoding and decoding time complexities. The reader is referred to [49] for more details about such codes, also known as fountain codes, and to [55] for a practical evaluation.

2.2.2 ROME

ROME [27] stands for Rateless Online MDS Erasure codes. These erasure codes were developed for wireless data broadcasting, and can dynamically adapt the amount n of generated coded symbols to the broadcasting conditions. They are built upon dedicated hardware to improve the overall encoding and decoding performance of classic Reed-Solomon (RS) codes [50], the most well-known class of MDS codes. ROME erasure codes are constructed over binary finite fields, the encoding and decoding algorithms are based on a Vandermonde matrix, and, without dedicated hardware, they have the same encoding and decoding time complexities as classic RS codes: $\Theta(nk)$ and $\Theta(k^2)$, respectively. Albeit n being dynamic and taking any value up to the field size, their practical application without dedicated hardware is limited to small values of k and n: typically up to 255 over $\mathbb{F}_{2^8}$.

2.2.3 Didier's

Didier [17] proposed encoding and decoding algorithms for RS codes over the binary finite field $\mathbb{F}_{2^m}$ with, respectively, $\Theta(n \log n)$ and $\Theta\left(n \log^2 n\right)$ time complexities, where $n = 2^m$ and is fixed to the size of the binary finite field. Thereby, the finite field being used must be chosen taking into consideration not only the performance of the arithmetic operations but also the required values of k and n; otherwise, $2^m - n'$ symbols will be encoded but never used, where n' is the number of effectively required coded symbols. These erasure codes resort on Walsh transforms to perform fast polynomial evaluation (encoding) and interpolation (decoding). In practice, the field size is limited up to 2^{16} so that the elements are easily representable (two bytes) and the products can be performed via in-cache lookup tables.

2.2.4 Soro et al.'s

Soro et al. [57] presented encoding and decoding algorithms with $\Theta(n \log n)$ time complexity over a finite field $\mathbb{F}_p$, where p is a Fermat prime ($p = 2^{2^m} + 1$). These codes are fixed-rate since all coded symbols are generated at once. The NTTs over these finite fields are usually referred to as Fermat Number Transforms (FNTs), and have $\Theta(n \log n)$ time complexity. Multiplicative NTTs can be computed efficiently using the same principle as complex Fast Fourier Transforms (FFTs), but there is an n-th root of unity only if $n \mid p - 1$. Thus, n can take any power of two value up to 2^{2^m}, which is, at most, 65536 given that Fermat finite fields are only known for $m \in \{0, 1, 2, 3, 4\}$. Arithmetic operations are efficient on current CPUs because they are regular integer modular arithmetic operations (mod p), but one element, 2^{2^m}, is not representable using 2^m bits.

2.2.5 Lin et al.'s

Lin et al. [38] proposed a non-standard polynomial basis to develop an $\Theta(n \log k)$ encoding and an $\Theta(n \log n)$ decoding algorithms over $\mathbb{F}_{2^m}$, where $n = 2^m$ and is fixed to the finite field size; k must be a power of two value and the rate $k/n \leq 0.5$. For a rate $k/n \geq 0.5$, and k taking also a power of two value, Lin et al. [37] recently proposed a new decoding algorithm with $\Theta\left(n \log(n-k) + (n-k) \log^2(n-k)\right)$ time complexity. As for Didier's work, the finite field must be chosen taking into consideration both the performance of the arithmetic operations and the required values of k and n, in order to minimize the amount of symbols encoded but never used. Their practical application is also limited up to 2^{16} for the same reasons.

2.2.6 Summary

Table 2.2 summarizes and compares the MDS erasure codes previously described taking into consideration the underlying finite field, the encoding and decoding time complexities without considering dedicated hardware implementations, the

Table 2.2: *Comparison of MDS erasure codes more suitable for P2P file sharing.* The comparison considers the underlying finite field, encoding and decoding time complexities (without considering any dedicated hardware), rate fixedness, and the domain of the number n of generated coded symbols.

Work	Finite Field	Time Complexity		Rateless	Number of Coded Symbols
		Encoding	Decoding		
ROME	$\mathbb{F}_{2^m}$	$\Theta(nk)$	$\Theta(k^2)$	✓	$1 < n \leq 2^m$
Didier	$\mathbb{F}_{2^m}$	$\Theta(n \log n)$	$\Theta(n \log^2 n)$	✗	$n = 2^m$
Soro et al.	$\mathbb{F}_{2^m+1}$	$\Theta(n \log n)$	$\Theta(n \log n)$	✗	$\{n : n \mid 2^{2^m}\}$
Lin et al.	$\mathbb{F}_{2^m}$	$\Theta(n \log k)$	$\Theta(n \log^2 n)$	✗	$n = 2^m$

rate fixedness, and the domain of the number n of erasure coded symbols that can be generated. ROME is the only rateless construction, but has quadratic decoding time complexity. Didier and Lin et al. proposed decoding algorithms to achieve $\Theta\left(n \log^2 n\right)$ time complexity, but n must be set to the size of the finite field. Soro et al. proposed encoding and decoding algorithms with $\Theta(n \log n)$ time complexity, the arithmetic operations over Fermat finite fields are efficient on modern CPUs, and k and n can take a larger and more flexible set of values. Nevertheless, their erasure codes are fixed-rate since all erasure coded symbols have to be generated at once.

Despite their merits, these works either proposed fixed-rate constructions or have high encoding and decoding time complexities. The practical use of those constructed over binary finite fields is limited up to 2^{16} because multiplications are likely to be performed using lookup tables, given that carry-less multiplication is not as efficient on current CPUs and large lookup tables severely degrade performance. Soro et al. proposed a construction with the lowest overall time complexity, but its practical use is also limited to the same level because the largest known Fermat finite field has $2^{16} + 1$ elements and n must take a power of two value.

2.3 ns-3 IPv4 Routing Protocols

In the context of ns-3, a routing protocol is simply a class or module that provides a route for outgoing packets and is able to forward, if needed, incoming packets. Unlike the canonical definition of routing protocols, which implies periodically and explicitly sending protocol messages, an ns-3 routing protocol may populate the routing tables using only global knowledge of the available network links without sending any protocol messages at all.

Section 2.3.1 provides a brief background on how custom ns-3 routing protocol can be implemented, and on how packets flow through the network and traverse the network stack. Sections 2.3.2 through 2.3.5 present existing ns-3 IPv4 routing protocols for infra-structured networks, such as the Internet, readily available as of version 3.22. The ns-3 routing protocols are compared in Section 2.3.6.

The description provided for each ns-3 routing protocol only considers unicast routing (multicast is not available over the Internet), and focus on the time complexity of sending an IP packet from source to destination as a function of both the number h of hops and the number n of routing table entries.

2.3.1 Background

ns-3 simulated nodes are interconnected by one or more `NetDevices` communicating over their respective channels. Packets flow through these channels and, as they traverse the network stack, the actual packet formats are provided to each layer, mimicking real networks. The network stack is configurable per node and, at node creation, it only includes the layers below the IP layer.

`InternetStackHelper` helper provides a simple interface to install the IP stack, and to set the ns-3 routing protocol that is to be used. Custom ns-3 routing protocols are implemented by deriving from `Ipv4RoutingProtocol` or `Ipv6RoutingProtocol` classes, respectively for IPv4 and IPv6[1], and must implement, at least, `RouteInput` and `RouteOutput` methods. `RouteInput` method returns a boolean indicating whether it takes responsibility for forwarding or delivering the packet; if so, one of four callbacks, with self-descriptive names, can be invoked to forward or deliver the packet: `ErrorCallback`, `UnicastForwardCallback`, `MulticastForwardCallback`, and `LocalDeliverCallback`. `RouteOutput` method is used by transport protocols to retrieve the route (`Ipv4Route` object) for outgoing packets, if any. These functions are executed for each incoming or outgoing packet, therefore, in particular for large-scale P2P file sharing simulations where a large amount of packets is exchanged between a large number of peers, they may represent a considerable share of time required to run a simulation (simulation time).

2.3.2 Ipv4ListRouting

`Ipv4ListRouting` is an ns-3 meta routing protocol that serves the sole purpose of combining different ns-3 routing protocols in a prioritized list. The ns-3 routing protocols are invoked in order until an incoming packet is handled (`RouteInput` invocation returns *true*) or a route is returned for an outgoing packet (through the invocation of `RouteOutput`). I.e., the next ns-3 routing protocol on the list is invoked only if the previous one either does not handle the incoming packet or does not provide a route for the outgoing packet. The `InternetStackHelper` helper installs by default an `Ipv4ListRouting` enclosing both `Ipv4StaticRouting` and `Ipv4GlobalRouting`, having `Ipv4GlobalRouting` higher priority than `Ipv4StaticRouting`.

2.3.3 Ipv4StaticRouting

`Ipv4StaticRouting` provides a basic set of methods for setting static unicast and multicast routes. Though it can be used as a standalone ns-3 routing protocol, it was

[1] Internet Protocol version 6.

designed to be inserted into an `Ipv4ListRouting` to complement other ns-3 routing protocols. `Ipv4StaticRouting` is usually used as a lower priority ns-3 routing protocol for setting default routes. The `RouteOutput` and `RouteInput` functions sequentially check each routing table entry until a matching entry with 32 bits mask is found or all entries have been checked. Thus, using `Ipv4StaticRouting`, sending an IP packet from source to destination has $\Theta\left(n_0 + n_1 + ... + n_h\right) \approx \Theta\left(\bar{n}\left(h+1\right)\right)$ time complexity, where n_i is the number of routing table entries at node i, and $\bar{n}$ is their average, being the source node 0 and the destination node h.

2.3.4 IPV4GLOBALROUTING

`Ipv4GlobalRouting` is the actual default ns-3 routing protocol, as it is the one with highest priority installed within `Ipv4ListRouting`. It walks the simulation topology, and populates node's routing tables with the routes returned by a shortest path algorithm. `Ipv4GlobalRoutingHelper` helper provides two class methods, `PopulateRoutingTables` and `RecomputeRoutingTables`, to populate the routing tables from scratch, and to update the routing tables, respectively. Thus, `Ipv4GlobalRouting` is easy to use and, given that both methods can be executed anytime during simulation, also supports runtime topology changes. Its `RouteOutput` and `RouteInput` functions sequentially check each routing table entry and collect all matching entries, if any. Thus, `Ipv4GlobalRouting` is able to send an IP packet from source to destination with $\Theta\left(n_0 + n_1 + ... + n_h\right) \approx \Theta\left(\bar{n}\left(h+1\right)\right)$ time complexity.

2.3.5 IPV4NIXVECTORROUTING

`Ipv4NixVectorRouting` [52] is a more efficient version of global routing that stores source routes in packet headers. The source node adds a set of bits to packet headers that indicate the interface that is to be used at each node to reach destination. The number of bits read by each node is the amount of bits required to represent all of its interfaces, being its value the interface index within the node. `Ipv4NixVectorRouting` enables runtime topology changes by recomputing the source routes and flushing caches either implicitly, when interface changes occur, or explicitly, by invoking `FlushGlobalNixRoutingCache` class method. Its `RouteOutput` function retrieves the path to a given destination in $\Theta\left(\log n\right)$, and its `RouteInput` function forwards packets in $\Theta\left(1\right)$. Therefore, an IP packet flows from source to destination with $\Theta\left(\log n_0 + h\right)$ time complexity.

2.3.6 SUMMARY

Table 2.3 compares the packet forwarding performance of ns-3 routing protocols considering the time complexities of `RouteOutput` and `RouteInput` functions, and its overall time complexity. `Ipv4StaticRouting` and `Ipv4GlobalRouting` have equivalent time complexities: $\Theta\left(n\right)$ for `RouteOutput` and `RouteInput` functions, and $\approx \Theta\left(\bar{n}\left(h+1\right)\right)$ for sending IP packets through the path. `Ipv4NixVectorRouting`

Table 2.3: *Comparison of existing ns-3 IPv4 routing protocols.* The comparison considers the time complexities of `RouteOutput` and `RouteInput` functions, and of sending an IP packet from source to destination as a function of both the number h of hops and the number n of routing table entries; n_i is the number of routing table entries at node i (source is 0 and destination is h), and $\bar{n}$ their average.

ns-3 Routing Protocol	Time Complexity		
	RouteOutput	RouteInput	Source $\rightarrow$ Destination
Ipv4StaticRouting	$\Theta(n_0)$	$\Theta(n_i)$	$\approx \Theta(\bar{n}(h+1))$
Ipv4GlobalRouting	$\Theta(n_0)$	$\Theta(n_i)$	$\approx \Theta(\bar{n}(h+1))$
Ipv4NixVectorRouting	$\Theta(\log n_0)$	$\Theta(1)$	$\Theta(\log n_0 + h)$

is able to decrease the time complexity of sending IP packets to $\Theta(\log n + h)$ by improving the time complexities of both `RouteOutput` and `RouteInput` functions: respectively, $\Theta(\log n)$ and $\Theta(1)$.

`Ipv4StaticRouting` and `Ipv4GlobalRouting` are akin performance-wise, but the former requires explicit population of routing tables while the latter provides a helper to populate the routing tables automatically. `Ipv4NixVectorRouting` is able to significantly decrease the time complexity of sending IP packets, although in a non-standard way (adding bits to packet headers). Still, the cost of updating the routing tables may be significant for frequent topology changes, and each packet requires $\Theta(\log n)$ operations before reaching the destination.

2.4 Conclusions

This chapter presented the related work for P2P file sharing systems providing privacy through plausible deniability, for MDS erasure codes more suitable for P2P file sharing, and for existing ns-3 IPv4 routing protocols as of version 3.22 of ns-3 network simulator.

Privacy-preserving systems require peers to advertise, either fully or partially, what they have downloaded, and an attacker may take advantage of this information to: narrow or even void plausible deniability; identify user content interests; obtain a proof that the user is able to access a content either entirely or in part. Such proof may be used, e.g., to held a law-abiding user legally liable for copyright infringement as a result of, unknowingly and unwillingly, downloading a single or a few chunks of an illegal content.

The available MDS erasure codes are either fixed-rate or present high encoding and decoding time complexities, and are therefore not able to cope with the P2P file sharing dynamics. Their practical application is also limited to about 2^{16}: those constructed over binary finite fields are limited by the multiplication operations, which are typically performed using lookup tables; those constructed over Fermat finite fields are limited by the finite field size.

Simulating large-scale P2P file sharing networks with thousands of peers requires considerable computational resources as just forwarding packets from source to destination requires at least $\log n$ operations, where n is the number of routing tables entries; for `Ipv4NixVectorRouting`, n is basically the number of peers (number of possible distinct destinations).

3 Mistrustful P2P Model

> *Law-abiding citizens value privacy. Terrorists require invisibility. The two are not the same, and they should not be confused.*
>
> **— Richard Perle**

This chapter describes the Mistrustful P2P model, and it is structured as follows. Section 3.1 provides an overview of the model and its main building blocks – content interest disguise and mistrustful sharing – to best describe how the problem this work aims to solve is addressed. Section 3.2 discusses the peer roles and their sharing behavior, the content sharing process, and the role of mistrustful sharing on it. The attack model considered is defined in Section 3.3. Sections 3.4 through 3.8 characterize the instantiations of each one of the mechanisms used on the evaluation of the Mistrustful P2P model.

3.1 Overview

The Mistrustful P2P model is built on the concept of mistrusting all the entities participating in the P2P network, hence its name, and therefore users are not required to establish any trust links in order to participate in the content sharing. It relies on two main building blocks – content interest disguise and mistrustful sharing – and aims at hiding user content interests through plausible deniability, in untrusted P2P networks, while overcoming the main limitations of akin P2P file sharing systems. These limitations can be summarized as follows: 1) peers are required to advertise what they download enabling passive attacks; 2) protection against active attacks is only achieved by introducing either trust requirements or considerable network overhead; 3) the privacy protection against both passive and active attacks is probabilistic; 4) legitimate users may be held legally liable for, unknowingly and unwillingly, downloading or relaying traffic of illegal contents.

Content interest disguise, as in other privacy-preserving P2P systems, hides user content interests through plausible deniability. The user content interests are hidden by downloading both contents that the user is interested in (genuine) and additional contents of no interest to the user (cover), as long as they cannot be distinguished. Registering at a tracker and joining a swarm no longer represents interest in that content, and user content interests can no longer be identified by monitoring just a small fraction of the network [11]. A content interest disguise scheme selects the set of cover contents, how much of each one to download, and may impose constraints

to the content sharing in order to hide user content interests. The evaluation of the Mistrustful P2P model is centered on the feasibility of its novelty, the mistrustful sharing building block, and thereby the proposal of a content interest disguise scheme is out of the scope of this work.

The mistrustful sharing building block enables the user to configure the required trade-off between privacy and performance by defining the size c of the largest colluding group to be protected against and the minimum amount m of chunks that are to be downloaded per cover content (minimum network disguise overhead), where $c \leq m < k$ for any content partitioned into k chunks so that there is no proof of full content download (proof of download). Proof of access to content data (proof of access) is avoided by encoding contents in a way that only enables decoding after full download. The mistrustful sharing building block is composed of two core mechanisms – erasure coding and disclosure constraint –, and three supporting mechanisms – block selection, request backoff, and peer selection. It enables the Mistrustful P2P model to overcome limitations 1) to 4) as follows.

Peers avoid advertising what they download by requesting from other peers random chunks, thus defeating passive attacks of any size; the chunk request process and the messages exchanged are described in Section 3.2. Attackers have then to engage in the content sharing because the chunks owned or missing are only implicitly disclosed to other peers while sharing. Active attacks, of a size up to c, are defeated by constraining the amount of chunks disclosed to any set of c peers to be, at most, m: an attacker is not able to distinguish cover contents from genuine ones because at least m chunks are downloaded of each content. As so, user content interests are deterministically hidden. For legitimate users, legal liability is prevented by not requiring peers to relay traffic on behalf of other peers, by never fully downloading cover contents (their data is never accessible), and by avoiding both proof of access and proof of download for any attack of a size up to c. The protection of user content interests and the user liability are discussed in more detail in Section 6.1 (Security Analysis), which also describes the countermeasures employed against common attacks.

The erasure coding mechanism is the core mechanism used to enable the probability of randomly retrieving a chunk to become significant, and to enable decoding only after full download. Considering a content divided into k chunks, uniformly distributed across the network, the probability of retrieving the last chunk is just $1/k$. The erasure coding mechanism generates a set of n erasure coded chunks (blocks), from a set of k chunks, so that any subset of k' blocks enables to retrieve the content, where $k' = k\,(1 + \epsilon(k))$ and $\epsilon(k)$ is the erasure coding overhead. If the n blocks are also uniformly distributed across the network, the probability of retrieving the last required block increases to $1 - {}^{(k'-1)}/_{n}$. As an example, for $k = 20$, $n = 5 \cdot k = 100$, and $\epsilon(k) = 0$ (optimal erasure code), the probability increases from 5% to 81%.

The disclosure constraint mechanism is the core mechanism used to ensure that no more than m blocks are disclosed (requested or shared) to any set of c peers (largest colluding group considered), and thus prevents proof of download ($m < k$). It also contributes to the reduction of the overhead due to cover downloads because only m blocks need to be downloaded per cover content in order to avoid the identification of genuine downloads among cover downloads. This mechanism enables the disclosure of at least one block to each peer ($m \geq c$), and, on average, m/c blocks can be requested from each peer. Reducing the amount of available block requests by either increasing c or decreasing m may impact the overall performance.

The remaining three mechanisms – block selection, request backoff, and peer selection – are defined to enable the evaluation of the Mistrustful P2P model, and to ensure that contents are timely downloaded. The block selection mechanism determines which block is to be offered to a given requesting peer, affecting the distribution of blocks among peers and therefore the probability of retrieving a useful block (innovative block). The request backoff mechanism determines the delay between block requests aiming at maximizing the amount of useful blocks that can be obtained from the available block requests in the shortest time frame. With the same aim, the peer selection mechanism selects a peer to which a block request will be sent.

The instantiation provided for each mechanism, described in Sections 3.4 through 3.8, is the one used to evaluate the Mistrustful P2P model in Section 6.2 (Performance Evaluation). Due to the large number of variables and factors at play, it is considered the impact of those that are expected to change more often – c and m, content size, peer arrival rate, and number of seeders –, and of cover downloads on the average download bitrate and on the download completion ratio. For the sake of clarity and tractability, other factors and variables such as inter-content relations, incentives to share, parallel chunk requests, and Internet connection heterogeneity are not considered. The evaluation aims at demonstrating the feasibility of the Mistrustful P2P model rather than at optimizing its overall performance, i.e., it aims at demonstrating that peers are able to timely download contents without advertising what they download.

In sum, the mistrustful sharing building block reinforces the content interest disguise because the distinction between genuine and cover contents is hardened by not disclosing what peers download or miss, and a larger set of cover contents can be used as they do not need to be fully downloaded. It prevents legitimate users from being held liable due to cover content and misleading content downloads. Cover contents are never fully downloaded to guarantee that the user has never access to their content data. Misleading contents may be fully downloaded, but there is no proof of access or proof of download for any attack of a size up to c.

Table 3.1: *Peer roles and sharing behavior on the Mistrustful P2P model.* A seeder has all data chunks and generates a new block (erasure coded chunk) for each incoming request; a commoner is a peer that may be still downloading enough blocks to reconstruct the content data and shares the blocks it has already downloaded. As so, a commoner never shares data chunks and only has access to the content data after fully downloading the content.

Peer Role	Data Chunks	Blocks	
	Owned	To Share	To Download
Seeder	All	Generated blocks	None
Commoner	None	Downloaded blocks	Enough useful blocks

3.2 Peer Roles and Content Sharing

Peers, per content, can take one of two roles depending on their privacy requirements and the way they contribute to the file sharing: *seeder* – peer having a content that wants to share, and willing to forgo its privacy –, or *commoner* – peer willing to participate in the content sharing if its privacy requirements can be met. A seeder may be the author or a party interested in publishing a content, and therefore does not require the concealment of user content interests. It generates a unique block (erasure coded chunk) for each request it receives, and only refuses to serve block requests if it has no resources available. On the other hand, a commoner does not generate new blocks, only shares them if the user content interests remain hidden, and only has access to the content data after fully downloading the content. A commoner keeps track of the blocks it shares with other peers both for privacy protection and to avoid offering an uploaded block twice to the same peer. It may refuse to serve block requests if it has no useful blocks to offer, due to resource or privacy constraints, or due to content interest disguise strategies. The commoner never discloses the reason behind refusal. Table 3.1 summarizes the peer roles and their sharing behavior.

The block download process on the Mistrustful P2P model differs considerably from the one on other P2P file sharing systems, given that peers do not advertise what they download. This process is summarized in Figure 3.1, where peer A is the

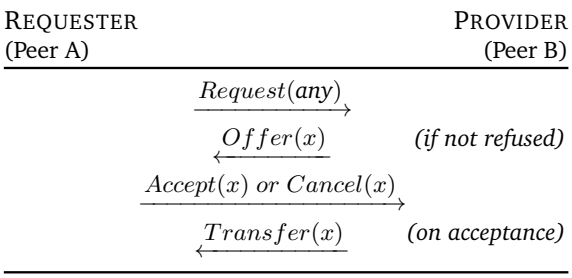


Figure 3.1: *Messages exchanged during block download process.* Peer A requests a random block from peer B, which either refuses the request or offers a block with $id\ x$. If the block is accepted by peer A, it will get transferred; otherwise, no transfer occurs to avoid unnecessary network overhead.

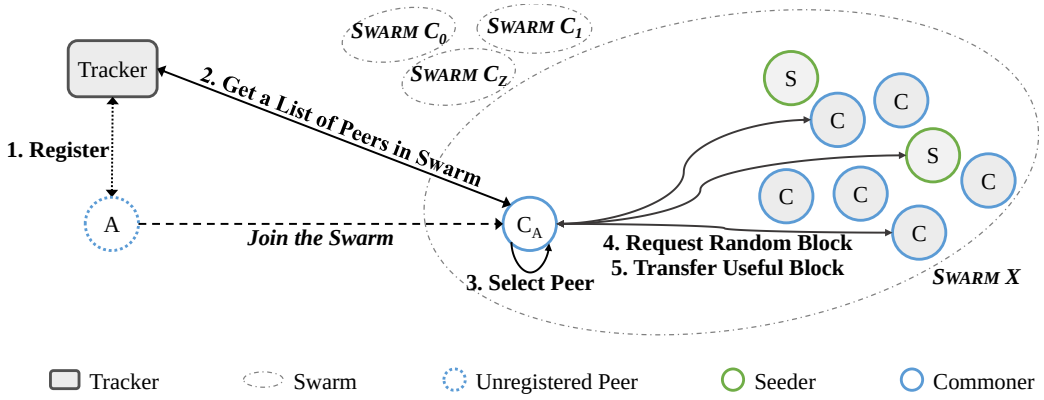


Figure 3.2: *Overview of the content download process on the Mistrustful P2P model.* First, peer A registers at the tracker to join the swarm of content X (swarm X) without disclosing what blocks it owns or misses; it also joins swarms C_0 through C_z to disguise user content interests. Then, it requests a list of peers (a subset) in the swarm from the tracker. Until download completion, and as long as the privacy requirements are met, peer A selects eligible peers in the swarm and requests random blocks from them. To prevent unnecessary network overhead, only offered blocks that are useful get transferred; otherwise, the block requests are canceled. The list of peers may be updated during content download, and there is no notification upon download completion.

requester and peer B is the provider, and works as follows. Peer A requests a random block from peer B, i.e., without providing the id of an intended block. Peer B then either refuses the request by simply ignoring it, and thus not disclosing the reason for refusal, or replies with the id of the block it is willing to share, which must be different from any other that may have been previously disclosed between them (both as requester and as provider). If peer B has offered a block, peer A sends a reply message either accepting the offered block, in order to start its download, or canceling the block request, in order to avoid unnecessary network overhead. Unless the block request has been canceled, peer B sends the block it has offered to peer A. The possible outcomes of a block request are further discussed in Section 3.7.

The content download process of the Mistrustful P2P model, depicted in Figure 3.2, consists in the following steps: 1) the peer registers at the tracker to join the swarms of both genuine and cover contents in order to disguise the content interests of the user, without disclosing its role; 2) it requests a list of peers (a subset) in the swarm from the tracker, which is not aware of each peer's role; 3) it selects peers from that list that cope with the user privacy requirements (eligible peers); 4) it requests random blocks from those peers in order to complete its download; 5) if the selected peers offer useful blocks, those blocks get transferred; otherwise, the requests are canceled and no transfer occurs. The list of peers may be updated during content download, and steps 3), 4) and 5) are repeated until download completion (genuine contents) or until at least m blocks have been transferred (cover contents). There is no notification from the commoner upon download completion.

The scope of action of the mistrustful sharing mechanisms entails mostly steps 3), 4) and 5). Their role on the content sharing is illustrated in Figure 3.3. The

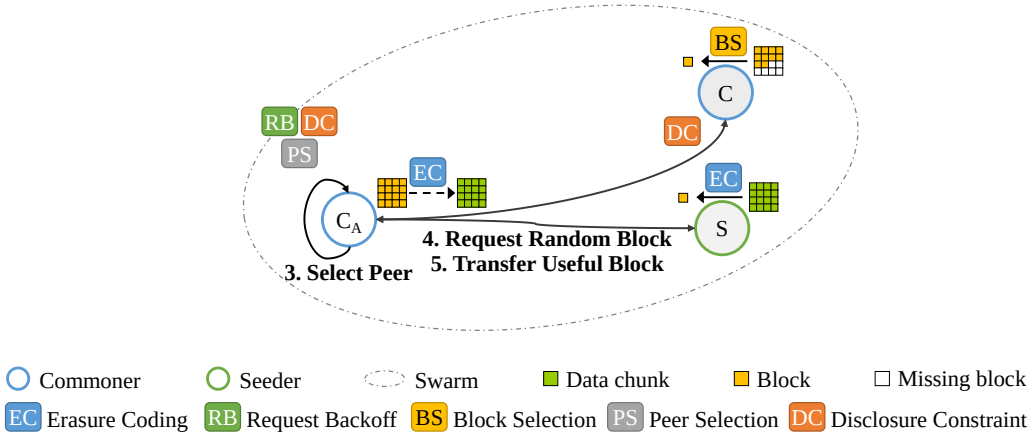


Figure 3.3: *Mistrustful P2P model mechanisms and their role in the content sharing.* The erasure coding mechanism is used by a seeder to generate new blocks, and by a commoner to retrieve the original data after fully downloading a content. The disclosure constraint mechanism determines if a block request can be sent to a peer or accepted by a given commoner. The block selection mechanism determines which block is to be shared, if any; the same block is never offered twice to the same peer. The peer selection mechanism selects an eligible peer to which a block request can be sent. The request backoff mechanism defines the delay between block requests.

erasure coding mechanism enables a commoner to retrieve the content data after fully downloading the content, and enables a seeder to generate a new block for each incoming request. The disclosure constraint mechanism determines if a block request can be sent to a peer or accepted by a given commoner. The block selection mechanism is used by the contacted commoner to determine which block is to be offered to the requesting peer, if any. The request backoff mechanism determines the delay between block requests of a commoner. The peer selection mechanism selects the peer to which the block request is sent.

3.3 Attack Model

It is assumed that an attacker might be any entity that participates in the system: a publisher, a tracker, a regular peer or a group of colluding peers. Being a participant of the system, it is considered that an attacker is able to engage in the content sharing as a commoner or as a seeder, to be a tracker, and to publish contents with misleading description. Also, it is considered that an attacker may coordinate a large number of peers that collude with each other (collusion attack) or assume multiple pseudonymous identities (Sybil attack), which is equivalent to a larger colluding group. External entities monitoring all traffic of a peer, such as ISPs, or controlling the whole network, such as governments, are not considered. Protection against link monitoring could be achieved by encrypting communications between peers, but requires key exchange and distribution mechanisms, which are out of the scope of this work.

The creation of large number of pseudonymous identities (Sybil attack) was first considered by Douceur [19], and is one of the most dangerous attacks that plague P2P networks [62]. Douceur [19] showed that, without trusted identity certification, Sybil attacks are always possible when considering realistic scenarios. In untrusted P2P networks, unlike collusion attacks, Sybil attacks can be mitigated without explicit information either by performing resource testing or by applying recurring costs and fees [62]. The reader is referred to [40] for a survey on existing approaches.

In the context of this work, collusion and Sybil attacks aim at increasing the amount of blocks disclosed by other peers to a single entity or to colluding entities, so that either content download can be proven or user content interests can be determined. The Mistrustful P2P model provides privacy protection against such attacks of a size up to c peers (colluding group), be it colluding peers, Sybil peers, or a combination of both. It requires $c \leq m < k$, and therefore m and k are upper bounds for c. On the one hand, despite increasing the minimum network disguise overhead, m can be increased as needed (up to k) given that it is user configurable. On the other hand, k is defined by the publisher of the content and cannot be changed. Thereby, in order to extend the privacy protection without increasing the size of the largest colluding group considered, c, the user is able to configure as single entities all the sets of peers that he considers, or suspects, to be colluding or to be Sybil peers.

Peers are identified by their public IP addresses because it is considered that IP addresses provide a more flexible resource testing that can be made harder to acquire than other resources such as human time, network bandwidth, computational power or storage capacity, which cannot be related. For privacy protection purposes, a set of peers whose IP addresses are considered to be related can be treated as a single peer (IP address aggregation), or multiple peers sharing a single IP address can be treated individually by considering the (IP, port) pair as the identifier (IP address multiplexing). The user is then able to configure how IP addresses are treated for privacy protection purposes, providing the flexibility to go as low as treating all (IP, port) pairs as unique identities, e.g. all peers behind NAT[1], up to treating all IP addresses of a given entity, colluding entities, city, country or any other set of IP addresses as a single peer. This also enables the user to configure the set of rules that are applied to peers using anonymous systems such as Tor, which can be IP address aggregation rules, IP address multiplexing rules or any combination of both.

3.4 Erasure Coding Mechanism

The erasure coding mechanism supports both MDS and non-MDS rateless erasure codes. It is designed to enable the probability of randomly retrieving a block

[1] Network Address Translation.

to become significant, and to enable decoding only after full download (avoid partial access). It enables the probability of randomly retrieving a block to become significant by generating a set of n blocks, from a set of k chunks, so that any subset of k' blocks enables to retrieve the content, where $k' = k\,(1 + \epsilon(k))$, $n > k$, and $\epsilon(k)$ is the erasure coding overhead. Decoding is enabled only after full download by either using a non-systematic erasure code that starts the decoding process only after receiving at least k' blocks or by encrypting the content such that decryption is only possible under the same conditions.

As discussed in Chapter 2, there is a trade-off between the encoding and decoding time complexities and the erasure coding overhead: decreasing the former is only possible by increasing the latter. The network is typically the most constrained P2P file sharing resource, not the CPU [36], and therefore MDS erasure codes $[\epsilon(k) = 0]$ are more suitable, especially those that are non-systematic and only enable decoding after receiving k blocks as additional computational overhead due to encryption is also avoided. Rateless erasure codes are also able to cope with the peer dynamics and to adjust to different content sharing conditions given that they do not impose any practical constraints to the amount n of generated blocks. Therefore, a rateless, MDS, and non-systematic erasure code with as low as possible encoding and decoding time complexities is desirable.

For evaluation purposes, the erasure coding mechanism uses the Storm erasure codes, a rateless MDS construction of RS codes developed for Mistrustful P2P, and presented in Chapter 4, with $\Theta\,(n \log k)$ encoding time complexity and $\min\left\{\Theta\,(n \log n)\,, \Theta\left(k \log^2 k\right)\right\}$ upper bound for decoding time complexity. These erasure codes are defined over the finite field $\mathbb{F}_{p^2}$, where p is a Mersenne prime $(p = 2^m - 1)$, and $n \leq 2^{m+1}$. Their performance does not impose any constraints to the content sharing, and they only enable decoding after full content download, thus avoiding partial access.

3.5 Disclosure Constraint Mechanism

The disclosure constraint mechanism enables the user to configure, per content, the required trade-off between privacy and performance by setting the size c of the largest colluding group to be protected against, and the minimum amount m of blocks that need to be downloaded per cover content. The size c of the largest potential attacker is defined by the number of unique peers (unrelated public IP addresses) that are controlled by a single group, either a single attacker or a group of colluding attackers. This mechanism ensures that cover and genuine content downloads cannot be distinguished by tracking their download progress because at most m blocks are disclosed to any set of c peers, and that no malicious peer can prove that a user downloaded a content or had access to its data $(m < k)$.

Finding the maximum intersection between the set of blocks disclosed to any set of c peers is an NP-hard problem [54], thus it was devised a conservative yet efficient algorithm to evaluate dynamically the number of blocks that can still be shared with a peer. The algorithm is divided into two main functions: one to update the counter of blocks disclosed to a peer (Function 1 – Update Blocks Disclosed), and the other to determine the number of blocks that can still be disclosed to a peer (Function 2 – Blocks to Disclose Left). The variables $commoners$ and $blocksDisclosed$ are respectively an array sorted by the number of blocks disclosed, and the maximum number of blocks disclosed to any set of c peers.

Function 1 Update Blocks Disclosed

```
function IncrementBlocksDisclosed(id)
    i ← commoners.getIndex(id)
    if invalidIndex(i) then                                   ▷ New.
        commoners.push(id)
        commoners.last.blocks ← 1
        i ← commoners.getIndex(id)
    else                                                      ▷ Known.
        commoners[i].blocks ← commoners[i].blocks + 1
        j ← i − 1
        while validIndex(j) do
            blocksI ← commoners[i].blocks
            blocksJ ← commoners[j].blocks
            if blocksI > blocksJ then                         ▷ Still unsorted.
                swap(commoners[i], commoners[j])
                i ← j
                j ← j − 1
            else                                              ▷ Sorted.
                break
            end if
        end while
    end if
    if i < c then                                             ▷ Changes on top c peers.
        blocksDisclosed ← blocksDisclosed + 1
    end if
end function
```

Function 1 receives as input the id of the peer to which one additional block was disclosed. If none has yet been disclosed, a new entry is created; otherwise, the entry is updated and, if needed, some elements are swapped to keep the array sorted. In each case, if the updated entry is on one of the top c positions, the maximum number of blocks disclosed is updated. Function 1 has linear time complexity.

Function 2 also receives as input the id of the peer. If the configured privacy requirements are not met (invalid), no blocks can be disclosed to any peer. If they are met, $left$ contains the number of blocks that can still be disclosed, ensuring that at least one block can be disclosed to each one of the top c peers; at most, $m-(c-1)$ blocks can be disclosed to a single peer. $left$ needs to be updated if there are already at least c peers and the peer referred by id is outside of that set. Function 2 runs in logarithmic time.

Function 2 Blocks to Disclose Left

```
function BlocksDiscloseLeft(id)
    if c > m or m ≥ k then                                  ▷ Invalid.
        return 0
    end if
    top ← min(c, commoners.length)                          ▷ Top peers.
    left ← m − blocksDisclosed − (c − top)
    i ← commoners.getIndex(id)
    if invalidIndex(i) then                                 ▷ New.
        if commoners.length ≥ c then
            left ← left + commoners[c − 1].blocks
        else
            left ← left + 1
        end if
    else                                                    ▷ Known.
        if i ≥ c then
            left ← left + commoners[c − 1].blocks
            left ← left − commoners[i].blocks
        end if
    end if
    return left
end function
```

3.6 Block Selection Mechanism

The block selection mechanism is used by commoners to determine which block is to be offered to a requesting peer. It plays an important role on how the blocks end up distributed across the network, affecting the probability of peers obtaining useful blocks. This mechanism ensures that no uploaded block is offered twice to the same peer, and determines when requests should be refused. The request refusal may be due to the lack of useful blocks to offer, due to resource or privacy constraints, or due to content interest disguise strategies.

Aiming at balancing the distribution of blocks across the network, each peer attributes a weight w_i to each block it owns. The weights are updated according to the perception of the peer about their availability, which is based on the acceptance or cancellation of the offered blocks. The blocks to offer are picked through a random weighted selection, and recently downloaded blocks start with a weight w_s. When a block is offered, if it is accepted, the weight is updated using an additive (aging) factor, w_λ; otherwise, the weight is updated using a multiplicative (replica control) factor, w_η. Thus, to ensure that the weight decrease on acceptance is never greater than the one on cancellation, the weight update is given by Equation 3.1.

$$w_i = \begin{cases} max\left(1, w_i - w_\mu\right), & \text{if it is accepted} \\ max\left(1, \left\lfloor \frac{w_i}{w_\eta} \right\rfloor\right), & \text{otherwise} \end{cases} \qquad \text{where } w_\mu = min\left(w_i - \left\lfloor \frac{w_i}{w_\eta} \right\rfloor, w_\lambda\right) \tag{3.1}$$

With the Mistrustful P2P model there is no need to suddenly terminate or remove downloads nor to stop sharing because the provided protection does not depend on the time a peer keeps sharing a content, as long as cover and genuine downloads are treated the same way. To evaluate the Mistrustful P2P model, it was considered $w_s = 100$, $w_\lambda = 5$, and $w_\eta = 2$ with the goal of favoring more recent blocks.

3.7 Request Backoff Mechanism

The request backoff mechanism determines the delay between block requests to help maximizing the amount of useful blocks that can be obtained from the available block requests in the shortest time frame. The mechanism identifies the set of peers to which block requests can be sent (eligible peers), and determines for how long no block requests should be sent. Therefore, as the former is a direct result of individual peer behavior and the latter depends on the swarm behavior, the backoff time is defined as a two-dimensional variable that has per peer and per swarm components. The peer backoff component provides the delay to return a peer to the set of eligible peers; the swarm backoff component provides the delay until the next block request. The actual backoff time is randomly generated within the interval $[0, b]$, where b is the calculated backoff time.

A block request has five possible outcomes: 1) refusal – the request is refused by the contacted peer (no offer); 2) cancellation – the request is canceled by the requester (duplicate block); 3) acceptance – the request is accepted and a block is downloaded; 4) interruption – the request is accepted but the download is interrupted; 5) disposal – no request is sent due to the lack of eligible peers. Refusal and disposal disclose no information about block ownership, but all the others do. Cancellation and acceptance reveal that both peers already own that block; interruption reveals that the contacted peer owns that block. Figure 3.4 illustrates each possible outcome.

Consider u and v to be respectively the number of consecutive refused requests and the number of consecutive requests that were either canceled or interrupted, and u_i and v_i the same variables but for a peer i; by consecutive it is meant until a request is accepted. Let μ, α, λ, β, η, and $\bar{\tau}$ be respectively the maximum backoff time, the base linear backoff time, the linear backoff time factor, the base exponential backoff time, the exponential backoff time factor, and the estimated average block transfer time. The backoff time b is then given by Equation 3.2.

$$b = min\left(\mu, \alpha + \lambda u + \beta\left(\eta^v - 1\right)\right) \tag{3.2}$$

For block requests that do not disclose block information, u, it is applied a linear increase, λ; for block requests that disclose block information, v, it is applied an exponential increase, η. The backoff time never exceeds μ. The peer and the swarm

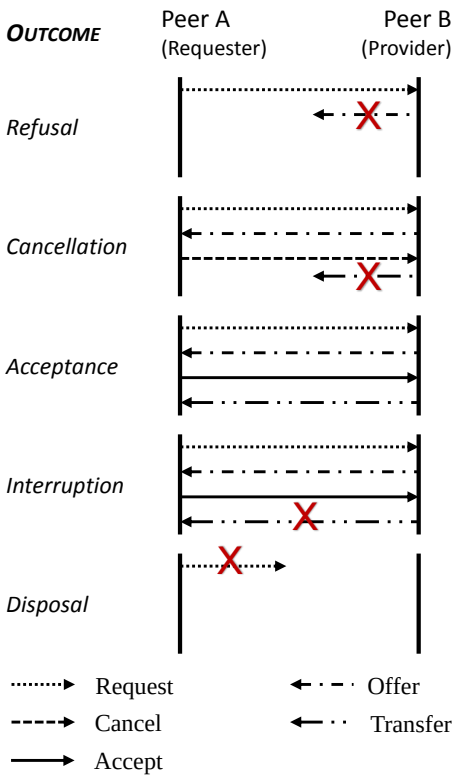


Figure 3.4: *Possible outcomes of a block request.* Refusal – the request is refused by peer *B* (no offer); cancellation – the request is canceled by peer *A* (no transfer); acceptance – the request is accepted and the block is transferred from peer *B* to peer *A*; interruption – the request is accepted but the block transfer is interrupted; disposal – no request is sent.

components are expressed using Equation 3.2, but the values of their variables may be different. Thus, the variables of each component are distinguished by a p subscript (peer) and an s subscript (swarm). For the swarm component only, when there are no eligible peers $(\forall_i, b_{p_i} > 0)$, $b_s = min\,(\lambda_p, min\,(b_{p_i}))$.

For the sake of clarity and tractability, the evaluation assumes no simultaneous block requests, homogeneous Internet connections, and a single content download. Therefore, it was considered $\alpha_p = 100$ ms, $\beta_p = \bar{\tau}/4$, $\lambda_p = \bar{\tau}/10$, $\eta_p = 2$, and $\mu_p = \frac{k\bar{\tau}c}{m}$ for the peer component. The per peer backoff time is a function of the block transfer time, and is instantiated such that the block requests to the same peer are, on average, 50 ms apart and such that, on average, m/c block requests can be sent to each peer during content download. m/c represents the configured protection and $k\bar{\tau}$ the minimum time required to download a given content. For the swarm component, it was considered $\alpha_s = 0$ ms, $\beta_s = \bar{\tau}/4\rho$, $\lambda_s = \bar{\tau}/16\rho$, $\eta_s = 2$, and $\mu_s = \bar{\tau}$, where ρ is the number of active peers. The swarm backoff time is a function of both block transfer time and the number of active peers, which can be obtained from the tracker(s). Given that the evaluation assumes no simultaneous block requests, there is no minimum delay required between consecutive requests, which can be, at most, the time required to download a single block.

3.8 Peer Selection Mechanism

The peer selection mechanism aims at selecting the eligible peer that has the highest probability of providing a useful block in less time. The set of eligible peers is constrained both by the disclosure constraint and request backoff mechanisms. The former provides, for a given content, the set of peers to which no further block requests can be sent to; the latter provides, for the same content, the set of peers that are ineligible for the moment, and when can the next block request be sent to each one of them.

For the sake of clarity and tractability, peers were selected randomly in order to avoid adding an additional factor into the evaluation. Non-uniform selection of peers is expected to impact on how the blocks end up distributed across the network, especially when considering heterogeneous Internet connections: peers with more available bandwidth are able to replicate more rapidly the blocks they own.

3.9 Conclusions

The Mistrustful P2P model aims at hiding user content interests, without having any trust requirements, and at preventing user legal liability in case of legitimate usage while enabling timely content downloads. It provides plausible deniability, ensures deterministic protection against attacks of a size up to a configured level, and enables the user to configure the required trade-off between privacy and performance. Its attack model considers that an attacker might be any entity that participates in the system, but not external entities monitoring all traffic of a peer (out of scope). Its two main building blocks are content interest disguise, and mistrustful sharing, being the former reinforced by the latter.

Content interest disguise, as in other privacy-preserving P2P systems, hides user content interests by downloading both genuine and cover contents. Yet, user content interests are deterministically hidden against any attack of a size up to c (size of the largest colluding group considered). The mistrustful sharing building block prevents user legal liability, and reduces network overhead while reinforcing content interest disguise. Legal liability is prevented by avoiding proof of access to content data, and by enabling peers to communicate through direct links. In turn, proof of access is avoided by avoiding partial access – chunks are encoded in a way that enables decoding only after full download –, and by avoiding proof of download – the downloaded chunks are not advertised, and the set of those implicitly disclosed while sharing is constrained. The network overhead due to cover contents can be reduced, up to a minimum amount m of chunks per cover content, and still ensure that they cannot be distinguished from genuine contents. Variables c and m impact the overall performance and thus can be configured by the user.

Five mechanisms are defined to be able to evaluate the Mistrustful P2P model, and to ensure that contents are timely downloaded: erasure coding, disclosure constraint, block selection, request backoff, and peer selection mechanisms. Peers avoid advertising what they own or miss by requesting random blocks, and the erasure coding mechanism enables the probability of randomly retrieving a useful one to become significant. The disclosure constraint mechanism thwarts active attacks by ensuring that no more than m blocks are disclosed (requested or shared) to any set of c peers, and thus cover and genuine content downloads cannot be distinguished by tracking their download progress because at most m blocks are disclosed to any set of c peers, and no malicious peer can prove that a user downloaded a content or had access to its data ($m < k$). The block selection mechanism determines which block is to be offered to a given requesting peer, affecting the distribution of blocks among peers and therefore the probability of retrieving a useful block. The request backoff mechanism determines the delay between block requests aiming at maximizing the amount of useful blocks that can be obtained from the available block requests in the shortest time frame. The peer selection mechanism selects a peer to which a block request will be sent to.

Peers, per content, can either be seeders or commoners. Seeders have a content to be shared, are willing to forgo their privacy, and provide a new block for each block request they receive; block requests are only refused if there are no resources available. Commoners are only willing to participate in the content sharing if user's privacy requirements can be met, do not generate new blocks, and only share them if user content interests remain hidden. They only have access to the content data after fully downloading the content, and keep track of what they share with other peers both for privacy protection and to avoid offering a block twice to the same peer. Block requests may be refused if there are no useful blocks to offer, due to resource or privacy constraints, or due to content interest disguise strategies, but the reason behind refusal is never disclosed.

The mistrustful sharing building block reinforces the content interest disguise because the distinction between genuine and cover contents is hardened by not disclosing what peers download or miss, and a larger set of cover contents can be used as they do not need to be fully downloaded. Cover contents are never fully downloaded to guarantee that the user has never access to their content data. Misleading contents may be fully downloaded, but user legal liability is prevented, against any colluding group of a size up to c, given that such attacker cannot prove neither the access nor the download of a content.

STORM ERASURE CODES

4

> *When it comes to privacy and accountability, people always demand the former for themselves and the latter for everyone else.*
>
> **— David Brin**

Erasure codes are a class of Forward Error Correction (FEC) codes for the Binary Erasure Channel (BEC), a channel in which the transmitted symbols are either correctly received or not received at all (erasure). Networking layers above the data link layer behave as an erasure channel since packets are either correct, and are delivered, or present errors, and are discarded. Therefore, the P2P file sharing overlay network behaves also as a binary erasure channel.

In the context of P2P file sharing, an erasure code enables the generation of a set of n blocks (erasure coded chunks) from a set of k chunks, $k < n$, so that any subset of $k\,(1 + \epsilon(k))$ blocks is enough to reconstruct the original chunks, where $\epsilon(k)$ is the erasure coding overhead. Erasure codes are usually classified according to three orthogonal properties: systematicity, rate fixedness, and coding overhead. If the input data chunks are also present in the set of output blocks, the erasure code is systematic; otherwise, it is non-systematic. If the amount n of blocks is fixed and needs to be known before encoding, the erasure code is fixed-rate. If new blocks can be dynamically generated as needed, the erasure code is rateless. Lastly, an erasure code is MDS (or optimal) if no additional blocks are required to reconstruct the original information $[\epsilon(k) = 0]$, or non-MDS otherwise $[\epsilon(k) > 0]$.

Storm erasures codes are a rateless MDS construction of Reed-Solomon codes developed for the Mistrustful P2P model that enable both systematic and non-systematic encoding. These erasure codes are MDS because the network is typically the most constrained P2P file sharing resource, rateless to be able to cope with and better adapt to the P2P file sharing dynamics, and enable non-systematic encoding in order to avoid partial access to the content data. Storm erasure codes are defined over the finite field $\mathbb{F}_{q=p^2}$, where p is a Mersenne prime $(p = 2^m - 1)$, and, although the construction of RS codes over such field has already been proposed [51], to the best of the author's knowledge, it was the first rateless construction (n can be increased in steps of k) to be proposed with $\Theta\,(n \log k)$ encoding time complexity and $\min\left\{\Theta\,(n \log n)\,, \Theta\left(k \log^2 k\right)\right\}$ decoding time complexity.

The remainder of this chapter is structured as follows. Section 4.1 provides an overview of Storm erasure codes. Sections 4.2 through 4.7, if required, provide a brief background before describing, respectively, the finite field over which Storm erasure codes are defined, the multiplicative NTT available over this field, the

multi-point polynomial algorithms, the mapping used to avoid ambiguity between 0 and $2^m - 1$, the encoding algorithm for both systematic and non-systematic codes, and the decoding algorithm. Section 4.8 presents the performance evaluation of Storm erasure codes along Soro et al.'s [57], the only related work with $\Theta(n \log n)$ encoding and decoding time complexities that admit any power of two value for n, up to the finite field size. Section 4.9 draws the main conclusions and future work.

4.1 Overview

Storm erasure codes are constructed akin to the original view of RS codes: construct a polynomial of degree $k-1$ over the finite field $\mathbb{F}_q$ and whose coefficients are the k data symbols to be transmitted, evaluate it at n points (code locators) to obtain the encoded symbols, and interpolate it to decode the k data symbols. For P2P file sharing, the aim is not at detecting and correcting errors but at enabling successful decoding of a content by downloading any k out of n blocks, and thus blocks are constructed as follows. For a content partitioned into k chunks, each of which is treated as a set of source (or data) symbols, construct a set of polynomials of degree $k-1$ whose coefficients are the source symbols at the same position of each chunk. Each block is then the set of symbols that result from evaluating each one of those polynomials at a given point (code locator).

Let $c_i = (s_{i,0}, \ldots, s_{i,d-1})$ be the set of the d source symbols composing chunk i, $s_j = (s_{0,j}, \ldots, s_{k-1,j})$ the source vector of size k composed of the source symbols at position j of each chunk, $s_j(x) = \sum_{\alpha=0}^{k-1} s_{\alpha,j} \cdot x^{\alpha}$ its polynomial representation, $b_l = (e_{l,0}, \ldots, e_{l,d-1})$ the set of d erasure coded symbols composing block l, and $e_j = (e_{0,j}, \ldots, e_{n-1,j})$ the encoded vector of size n composed of the erasure coded symbols at position j of each block, where $0 \leq i < k$, $0 \leq l < n$, and $0 \leq j < d$. Equation 4.1 then provides the matrix representation of a content (matrix $\mathbf{C}$) and the blocks generated from it (matrix $\mathbf{B}$). The rows of matrix $\mathbf{C}$ are the k chunks into which the content is partitioned, and its columns are the polynomials of degree $k-1$ being evaluated (source vectors). The rows of matrix $\mathbf{B}$ are the set of generated blocks, and its columns are the result of evaluating a given polynomial at n unique points (encoded vectors).

$$\mathbf{C} = \begin{array}{c} \begin{matrix} s_0 & \cdots & s_{d-1} \end{matrix} \\ \begin{bmatrix} s_{0,0} & \cdots & s_{0,d-1} \\ \vdots & \ddots & \vdots \\ s_{k-1,0} & \cdots & s_{k-1,d-1} \end{bmatrix} \end{array} \begin{matrix} c_0 \\ \vdots \\ c_{k-1} \end{matrix}, \quad \mathbf{B} = \begin{array}{c} \begin{matrix} e_0 & \cdots & e_{d-1} \end{matrix} \\ \begin{bmatrix} e_{0,0} & \cdots & e_{0,d-1} \\ \vdots & \ddots & \vdots \\ e_{n-1,0} & \cdots & e_{n-1,d-1} \end{bmatrix} \end{array} \begin{matrix} b_0 \\ \vdots \\ b_{n-1} \end{matrix} \tag{4.1}$$

The encoding process is defined by the transformation $\mathbf{C} \xrightarrow{\mathcal{T}} \mathbf{B}$, which is obtained by applying the transformation $(s_{0,j}, \ldots, s_{k-1,j}) \xrightarrow{\mathcal{F}} (e_{0,j}, \ldots, e_{n-1,j})$ over $\mathbb{F}_q^n$, with

$e_{l,j} = \sum_{\alpha=0}^{k-1} s_{\alpha,j} \cdot {x_l}^{\alpha} = s_j(x_l)$, to each one of the d columns. I.e., it is obtained by evaluating each one of the d polynomials of degree $k-1$, $s_j(x)$, at n unique points, x_l. Storm codes enable the transformation $\mathcal{T}$ to be performed in steps of k blocks.

The decoding process is defined by the inverse transformation $\mathbf{B} \xrightarrow{\mathcal{T}^{-1}} \mathbf{C}$, obtained by applying the inverse transformation $(e_{0,j}, \ldots, e_{n-1,j}) \xrightarrow{\mathcal{T}^{-1}} (s_{0,j}, \ldots, s_{k-1,j})$ over $\mathbb{F}_q^n$, with $s_j(x)$ as defined by Equation 4.2, to each one of the d columns. I.e., it is obtained by interpolating each one of the polynomials of degree $k-1$, $s_j(x)$, using at least k unique blocks, given that a polynomial of degree less than k is uniquely determined by any set of k unique pairs $(x_l, s_j(x_l))$. Let the Lagrange basis polynomial be $L(x) = \prod_{\alpha=0}^{k-1} (x - x_\alpha)$, the barycentric weights be $\omega_\alpha = \left(\prod_{\beta=0,\beta\neq\alpha}^{k-1} (x_\alpha - x_\beta)\right)^{-1}$, and $y_{\alpha,j} = e_{\alpha,j} \cdot \omega_\alpha$, then $s_j(x)$ is defined by Equation 4.2.

$$s_j(x) = \sum_{\alpha=0}^{k-1} e_{\alpha,j} \cdot \prod_{\beta=0,\ \beta\neq\alpha}^{k-1} \frac{x - x_\beta}{x_\alpha - x_\beta} = L(x) \cdot \sum_{\alpha=0}^{k-1} \frac{y_{\alpha,j}}{x - x_\alpha} \tag{4.2}$$

Storm erasure codes are defined over the finite field $\mathbb{F}_{q=p^2}$, where p is a Mersenne prime ($p = 2^m - 1$), also known as complex Mersenne finite fields. The encoding algorithm runs in $\Theta(n \log k)$ time, taking advantage of the multiplicative NTT that is known as Complex Mersenne Number Transform (CMNT) over these fields, and n can take any value multiple of k up to 2^{m+1}. The decoding algorithm runs in $\min\left\{\Theta(n \log n), \Theta\left(k \log^2 k\right)\right\}$ time, and uses internally either an interpolation method at n roots of unity that runs in $\Theta(n \log n)$ time or an interpolation method at k arbitrary points that runs in $\Theta\left(k \log^2 k\right)$ depending on the values k and n. Larger contents can be partitioned either into the same number of, but larger, chunks or into more chunks of the same size, but, unlike the latter, the former has only a linear increase on the encoding and decoding time complexities since it is just a matter of adding new columns (larger d) to matrices $\mathbf{C}$ and $\mathbf{B}$.

4.2 Finite Field

Finite fields, also known as Galois fields in honor of Évariste Galois, are fields with a finite set of elements. A prime field of p elements, $\mathbb{F}_p$, exists for any prime p, and its set of elements is $\{0, \ldots, p-1\}$. Arithmetic operations over prime fields are modular arithmetic operations (modulo p). Subtraction and division are defined in terms of addition and multiplication, respectively. For $a, b \in \mathbb{F}_p$, $a - b = a + (-b)$, and $a/b = a \cdot b^{-1}$, where $-b$ and b^{-1} are the unique elements that satisfy, respectively, $b + (-b) \equiv 0 \mod p$ (additive inverse), and $b \cdot b^{-1} \equiv 1 \mod p$ (multiplicative inverse). An extension field with $q = p^m$ elements, $\mathbb{F}_{p^m}$, is an extension over the prime field $\mathbb{F}_p$ whose elements are polynomials of degree $m-1$ with coefficients from $\mathbb{F}_p$. E.g., if $86_{10} = 01010110_2$ is an element of $\mathbb{F}_{2^8}$, its polynomial representation is $0x^7 + 1x^6 + 0x^5 + 1x^4 + 0x^3 + 1x^2 + 1x + 0 = x^6 + x^4 + x^2 + x$. Finite fields where $q = 2^m$ are named binary finite fields. Over extension fields the modulus is no longer

a prime p, but an irreducible polynomial of degree m. An irreducible polynomial is the equivalent of a prime number as it can only be divided by 1 and itself.

A primitive root of a finite field, r, is an element whose powers generate all non-zero elements and form a cyclic multiplicative group of $q-1$ elements, i.e., $\mathbb{F}_q = \{0, r^0, r^1, \ldots, r^{q-3}, r^{q-2}\}$. Unlike in $\mathbb{C}$, where there is always an n-th root of unity for any arbitrarily positive integer value of n, in $\mathbb{F}_q$ the equation $x^n = 1$ has not always n unique solutions. Instead, there is only an n-th root of unity if $n \mid q-1$. Unlike real arithmetic, finite field arithmetic involves only integer arithmetic, thereby, it introduces neither rounding nor precision errors. Moreover, the size of the result of an operation over a finite field is always the same because the result is itself an element of that field. These are the two main reasons for erasure codes arithmetic operations being performed over finite fields.

Storm erasure codes are constructed over the complex Mersenne finite field $\mathbb{F}_{q=p^2}$. The elements of this field can be defined as $\mathbb{F}_{p^2} = \{a + b\hat{\imath} \mid a, b \in \mathbb{F}_p\}$, where $\hat{\imath} = \sqrt{-1}$, given that the polynomial $x^2 + 1$ is always irreducible over $\mathbb{F}_p$, and that every irreducible quadratic polynomial over $\mathbb{F}_p$ must split over $\mathbb{F}_{p^2}$. Therefore, the existence of a root $\hat{\imath}$ for the polynomial $x^2 + 1$ is guaranteed in $\mathbb{F}_{p^2}$ [51], and the arithmetic operations can be performed over $\mathbb{F}_p$, being $\hat{\imath} \cdot \hat{\imath} \equiv -1 \mod p$. These operations can be efficiently computed on modern CPUs by optimizing the modular reduction, the evaluation of the additive inverse, and the evaluation of the multiplicative inverse. Addition and multiplication are regular integer operations to which modular reduction is applied. Subtraction and division are the same operations after evaluating, respectively, the additive and multiplicative inverses. For performance reasons, element 0 may be bitwise represented also as $2^m - 1$, given that $2^m - 1 \equiv 0 \mod p$.

The modular reduction over $\mathbb{F}_p$ is performed differently for sums and products, as the latter requires more computations. The result of a regular integer addition may require one additional bit to be represented – $\forall a, b \in \mathbb{F}_p, a + b \leq 2 \cdot (p-1)$ –, and the result of a regular product may require m additional bits to be represented – $\forall a, b \in \mathbb{F}_p, a \cdot b < 2^m \cdot (p-1)$. Given that $2^m \equiv 1 \mod p$, the modular reduction for addition is implemented by adding the value of the $(m+1)$-th bit to the value of the other m bits; the modular reduction for multiplication is implemented by adding the two m-bit values, and then performing a modular reduction as for addition. Thus, modular reduction only requires regular integer operations, and bitwise shifts.

The additive and multiplicative inverses over $\mathbb{F}_p$ can be computed as follows. The additive inverse in just an XOR with p, given that $-a \equiv p - a \mod p$ and $p = 2^m - 1$ (all m bits at one). The multiplicative inverse is the most costly operation and is computed using the extended Euclidean algorithm to determine the coefficients of Bézout's identity $ax + by = gcd(a, b)$, where gcd is the greatest common divisor. Equation 4.3 shows how the multiplicative inverse can be formulated as a Bézout's

identity. The multiplicative inverse a^{-1} is then coefficient x, which can be determined by the extended Euclidean algorithm.

$$\begin{aligned} a \cdot a^{-1} \quad &\mod p \equiv 1 \\ a \cdot a^{-1} + p \cdot k \quad &\mod p \equiv 1 \\ a \cdot x + p \cdot y \quad &\mod p \equiv gcd(a, p) \end{aligned} \tag{4.3}$$

Let $a, b, c, d \in \mathbb{F}_p$, and $z = a + b\hat{\imath}, w = c + d\hat{\imath} \in \mathbb{F}_{p^2}$, the arithmetic operations over $\mathbb{F}_{p^2}$ are performed as in $\mathbb{C}$ and are defined over $\mathbb{F}_p$ as follows: $z+w = (a+c)+(b+d)\hat{\imath}$, $z-w = (a-c)+(b-d)\hat{\imath}$, $z \cdot w = (a \cdot c - b \cdot d)+(a \cdot d + b \cdot c)\hat{\imath}$, and $z/w = (z \cdot \bar{w}) \cdot |w|^{-2}$, where $\bar{w} = c - d\hat{\imath}$ is the complex conjugate of w, and $|w|^2 = w \cdot \bar{w}$ is the square of its absolute value. The additive and multiplicative inverses over $\mathbb{F}_{p^2}$ are, respectively, $-z = (-a) + (-b)\hat{\imath}$, and $z^{-1} = \bar{z} \cdot \left(|z|^2\right)^{-1}$.

Complex Mersenne finite fields always have a multiplicative group of size 2^{m+1}, as $2^{m+1} \mid p^2 - 1$, whose root $r = 2^{2^{m-2}} + (-3)^{2^{m-2}}\hat{\imath}$ [16], and the components of the 8^{th} unity roots are fixed powers of two, only involving additions and circular shifts, enabling efficient radix-8 NTTs. Let $\lambda = 2^{(m-1)/2}$, the set of 8^{th} roots of unity is $\{1, -1, \hat{\imath}, -\hat{\imath}, \lambda(1+\hat{\imath}), \lambda(1-\hat{\imath}), \lambda(-1+\hat{\imath}), \lambda(-1-\hat{\imath})\}$ [51]. The inverse of a unity root z is its complex conjugate $(z \cdot z^{-1} = z \cdot \overline{z} = 1)$, and therefore it does not need to be computed through the extended Euclidean algorithm. Unlike Fermat fields, there is no known size limit for complex Mersenne fields, being $2^{74207281} - 1$ the largest known Mersenne prime.

4.3 Complex Mersenne Number Transform

Polynomial evaluation and polynomial interpolation at k points can be more efficiently computed by using the NTT and its inverse (INTT). Let $s_j(x)$ be a polynomial of degree $k-1$, then these transforms are defined over a finite field $\mathbb{F}_q$, respectively, as $S_{\alpha,j} = \sum_{\beta=0}^{k-1} s_{\beta,j} \cdot W_k^{\alpha \cdot \beta}$ and as $s_{\beta,j} = \frac{1}{k} \sum_{\alpha=0}^{k-1} S_{\alpha,j} \cdot W_k^{-\alpha \cdot \beta}$, where W_k is a k-th root of unity. At k unity roots, the NTT evaluates a polynomial of degree less than k in $N(k)$ time, and the INTT interpolates it also in $N(k)$ time, where $N(k)$ is $\Theta(n \log n \log\log n)$ for binary finite fields (additive NTT) [22] and $\Theta(n \log n)$ for non-binary finite fields (multiplicative NTT) [5]. At k arbitrary roots, the fast multi-point polynomial evaluation and interpolation algorithms achieve $M(k) \log k$ time complexity by performing $\log k$ steps of polynomial products, each step with time complexity equivalent to a single product of two polynomials of degree less than k, and where $M(k)$ is the time complexity of computing such polynomial product. The multi-point polynomial algorithms are described in Section 4.4.

The fast computation of a multiplicative NTT is achieved by decomposing the original sequence of computations into smaller subsequences while taking advantage of the symmetry and periodicity properties of the unity roots to reduce their overall count. The radix of the NTT refers to the number and size of the subsequences

that the original sequence is divided into at each stage, and the decimation is the way input data is combined, which can be either Decimation-In-Time (DIT) or Decimation-In-Frequency (DIF). The typical radixes are radix-2, radix-4, and radix-8 for decomposing a size k sequence into, respectively, 2, 4 and 8 equally sized subsequences at each stage, split-radix for decomposing a sequence of size k into one subsequence of size $k/2$ and two subsequences of size $k/4$, and mixed-radix when multiple radixes are used at different stages. The following paragraphs, although applicable to other NTTs, describe the CMNT through DIF because it is how the encoding algorithm is expressed in steps of k, and highlight the advantages of using a higher radix ρ over finite fields such as complex Mersenne finite fields, and for which there are efficient ρ-th roots of unity. The reader is referred to [12] for a through explanation of FFT algorithms, which are akin to NTT's.

The radix-2 DIF algorithm rearranges the CMNT into even- and odd-numbered computations, therefore a CMNT of a power of two size k, $S_{\alpha,j} = \sum_{\beta=0}^{k-1} s_{\beta,j} \cdot W_k^{\alpha \cdot \beta}$, is recursively expressed as given by Equation 4.4.

$$\begin{aligned} S_{2\alpha,j} &= \sum_{\beta=0}^{k-1} s_{\beta,j} \cdot W_k^{2 \cdot \alpha \cdot \beta} = \sum_{\beta=0}^{\frac{k}{2}-1} \left(s_{\beta,j} + s_{\beta+\frac{k}{2},j} \right) \cdot W_{\frac{k}{2}}^{\alpha \cdot \beta} \\ S_{2\alpha+1,j} &= \sum_{\beta=0}^{k-1} s_{\beta,j} \cdot W_k^{(2 \cdot \alpha+1) \cdot \beta} = \sum_{\beta=0}^{\frac{k}{2}-1} \left(\left(s_{\beta,j} - s_{\beta+\frac{k}{2},j} \right) W_k^{\beta} \right) \cdot W_{\frac{k}{2}}^{\alpha \cdot \beta} \end{aligned} \tag{4.4}$$

More generally, the radix-ρ DIF algorithm rearranges the CMNT of a power of two size k into groups of ρ subsequences of size k/ρ. Let $0 \leq \delta < \rho$ be the index of the subsequence being computed, then its expression is given by Equation 4.5.

$$S_{\rho \cdot \alpha+\delta,j} = \sum_{\beta=0}^{\frac{k}{\rho}-1} \left(\left(\sum_{\lambda=0}^{\rho-1} s_{\left(\beta+\lambda \cdot \frac{k}{\rho}\right),j} \cdot W_\rho^{\delta \cdot \lambda} \right) \cdot W_k^{\delta \cdot \beta} \right) \cdot W_{\frac{k}{\rho}}^{\alpha \cdot \beta} \tag{4.5}$$

The rearrangements made at each stage reverse the order of the output in groups of size ρ, and therefore, for any power of two ρ, the output indexes are reversed in groups of $\log_2(\rho)$ bits. The factors $W_k^{\delta \cdot \beta}$ and $W_\rho^{\delta \cdot \lambda}$ in Equation 4.5 are named *twiddle* factors, and the former are applied as regular multiplications, after combining the input data at each decomposition stage, because they depend on the current size of the CMNT ($0 \leq \beta < k/\rho$); the latter are ρ-th roots of unity that are applied as constants, when combining the input data, and therefore provide room for optimization as long as they can be computed more efficiently than regular multiplications. Unlike the FNT which supports only efficient radix-2 decimation, the CMNT supports efficient radix-8 decimation as there are 8 roots of unity over complex Mersenne finite fields that can be computed more efficiently than regular multiplications.

The CMNT implementation used to evaluate Storm erasure codes modifies the output order of radix-4 and radix-8 stages to match the one of radix-2 (single bit-reversed output indexes), and thus enable the mixed-radix algorithm to combine directly different radix algorithms. This way, the mixed-radix algorithm is able to easily select the most efficient combination to compute the CMNT, which should enable to take advantage of the radix-4 and radix-8 algorithms to compute the CMNT for any power of two size.

4.4 Multi-point Polynomial Algorithms

The CMNT and its inverse (ICMNT) enable to, respectively, evaluate and interpolate a polynomial at k roots of unity in $N(k)$ time. The multi-point polynomial algorithms extend polynomial evaluation and interpolation to k arbitrary points, but require $M(k) \log k$ time: $\log k$ stages of polynomial products, each stage with time complexity equivalent to a single product of two polynomials of degree less than k.

A polynomial $s_j(x)$ of degree $k-1$ can be represented in the more usual coefficient representation, $s_j(x) = \sum_{\alpha=0}^{k-1} s_{\alpha,j} \cdot x^{\alpha}$, or in the point-value representation, $\{(x_0, e_{0,j}), \ldots, (x_{k-1}, e_{k-1,j})\}$, where $e_{\alpha,j} = s_j(x_\alpha)$. The coefficient representation enables polynomial evaluation at k roots of unity in $N(k)$ time, but requires $\Theta(k^2)$ time to compute the product of two $k-1$ degree polynomials; the point-value representation enables the same polynomial product to be computed in $\Theta(k)$ time, but requires at least $2k-1$ unique point-value pairs for each polynomial so that the resulting polynomial can be accurately interpolated. The CMNT and the ICMNT enable to convert from one representation into the other, at k roots of unity, in $N(k)$ time. The product of two polynomials of degree $k-1$ is then computed in three steps: 1) convert both polynomials into the point-value representation using two CMNTs of size $2k$; 2) multiply the $2k$ values; 3) convert back to the coefficient representation using one ICMNT of size $2k$. Thus, although with a larger hidden constant, $M(k)$ has the same complexity as $N(k)$: assuming $N(2k) \approx 2 \cdot N(k)$, $M(k) \approx 6 \cdot N(k)$ given that the linear cost of the multiplications has negligible impact on the overall time complexity.

The multi-point evaluation algorithm is built upon the fact that polynomial evaluation can be expressed as a modular reduction: $s_j(x_\alpha) = s_j(x) \mod (x - x_\alpha)$. Polynomial modular reduction of two polynomials $s_j(x)$ and $p(x)$ of degree $k-1$ can be computed in $M(k)$ time, as it is a regular polynomial product once the inverse of polynomial $p(x)$ is determined, which takes also $M(k)$ time [15]. Let $p_{1,0} = \prod_{\alpha=0}^{\frac{k}{2}-1}(x - x_\alpha)$, and $p_{1,1} = \prod_{\alpha=\frac{k}{2}}^{k-1}(x - x_\alpha)$. Then, define $r_0 = s_j(x) \mod p_{1,0}$, and $r_1 = s_j(x) \mod p_{1,1}$. Since $r_0(x_\alpha) = s_j(x_\alpha)$, $0 \leq \alpha < k/2$, and $r_1(x_\alpha) = s_j(x_\alpha)$, $k/2 \leq \alpha < k$, the evaluation of $s_j(x)$ at k arbitrary points takes $\sigma = \log_2(k)$ recursive stages. At each stage u, $1 \leq u \leq \sigma$, 2^u polynomial modular reductions are performed using degree $(k/2^u) - 1$ reducing polynomials in $M(k)$ time: $2^u \cdot M(k/2^u) \approx M(k)$.

The set of polynomials $p_{u,v} = \prod_{\alpha=0}^{k'-1}(x - x_{v \cdot k' + \alpha})$ can be precomputed, as well as their inverses, given that it only depends on the points at which the evaluation took place, where $k' = k/2^u$ and v is the index of a reducing polynomial at stage u, $0 \leq v < 2^u$. If multiple polynomials are evaluated at the same points, this set only needs to be calculated once, which takes $M(k)\log(k)$ time using the relations $p_{\sigma,v} = (x - x_v)$ and $p_{u,v} = p_{u+1,2v} \cdot p_{u+1,2v+1}$. Therefore, multi-point evaluation algorithm runs in $M(k)\log(k)$, and is described in pseudo-code by Algorithm 3.

Algorithm 3 Multipoint Evaluation

(Pre)compute all $p_{u,v}$

function EVALUATOR(s, k, v) ▷ Evaluate $s_j(x)$ at k arbitrary points.
  if $k = 1$ **then return** s
  end if
  $u = \sigma - \log_2(k) + 1$
  $r_0 = s \mod p_{u,v}$, $r_1 = s \mod p_{u,v+1}$
  return EVALUATOR(r_0, $k/2$, $2v$), EVALUATOR(r_1, $k/2$, $2(v+1)$)
end function

$s(x_0), \ldots, s(x_{k-1}) =$ EVALUATOR($s, k, 0$)

The polynomial interpolation at k arbitrary points is akin to the converse of the multipoint evaluation. Let the Lagrange basis polynomial be $L(x) = \prod_{\alpha=0}^{k-1}(x - x_\alpha)$, $L'(x)$ its derivative, and the barycentric weights be $\omega_\alpha = \left(\prod_{\beta=0,\beta\neq\alpha}^{k-1} x_\alpha - x_\beta\right)^{-1}$. Computing the barycentric weights at k arbitrary points is a multi-point evaluation of $L'(x)$, given that $\omega_\alpha = (L'(x_\alpha))^{-1}$, and takes $M(k)\log(k)$ time. $L(x)$ can be computed through the product of $p_{1,0}$ and $p_{1,1}$ polynomials in $M(k)$ time. As for multi-point evaluation, if multiple polynomials are interpolated at the same k points, $L(x)$, $L'(x)$ and ω_α, $0 \leq \alpha < k$, only need to be calculated once. Let $y_{\alpha,j} = s_j(x_\alpha) \cdot \omega_\alpha$, Lagrange's interpolation formula can be rewritten as given by Equation 4.6, where $s_j(x)$ is the polynomial being interpolated.

$$s_j(x) = \sum_{\alpha=0}^{k-1} s_j(x_\alpha) \cdot \prod_{\substack{\beta=0 \\ \beta\neq\alpha}}^{k-1} \frac{x - x_\beta}{x_\alpha - x_\beta} = \sum_{\alpha=0}^{k-1} y_{\alpha,j} \cdot \prod_{\substack{\beta=0 \\ \beta\neq\alpha}}^{k-1} (x - x_\beta) \tag{4.6}$$

Let $r_0 = \sum_{\alpha=0}^{\frac{k}{2}-1} y_{\alpha,j} \cdot \prod_{\beta=0,\ \beta\neq\alpha}^{\frac{k}{2}-1} (x - x_\beta)$, and $r_1 = \sum_{\alpha=\frac{k}{2}}^{k-1} y_{\alpha,j} \cdot \prod_{\beta=\frac{k}{2},\ \beta\neq\alpha}^{k-1} (x - x_\beta)$. Observing that $s_j(x) = r_0 \cdot p_{1,1} + r_1 \cdot p_{1,0}$, a recursive algorithm can be devised that requires $\sigma = \log_2(k)$ stages, and thus runs in $M(k)\log(k)$ time. At each stage u, $1 \leq u \leq \sigma$, 2^u polynomial products of degree $(k/2^u) - 1$ polynomials are performed in approximately $M(k)$ time: $2^u \cdot M(k/2^u) \approx M(k)$. The multi-point polynomial interpolation algorithm is described in pseudo-code by Algorithm 4. The reader is referred to [15] for a thorough explanation of multi-point evaluation and interpolation algorithms.

Algorithm 4 Polynomial Interpolation at Arbitrary Points

(Pre)compute all $p_{u,v}$
Compute all y_α ▷ $y_{\alpha,j} = s_j(x_\alpha) \cdot w_\alpha$

function INTERPOLATOR(y, k, v) ▷ Interpolate $s_j(x)$ at k arbitrary points.
  if $k = 1$ **then return** y_0
  end if
  $u = \sigma - \log_2(k) + 1$
  $r_0 = \text{INTERPOLATOR}(\left\{y_0, \ldots, y_{k/2-1}\right\}, k/2, 2v)$
  $r_1 = \text{INTERPOLATOR}(\left\{y_{k/2}, \ldots, y_{k-1}\right\} k/2, 2(v+1))$
  return $r_0 \cdot p_{u,v+1} + r_1 \cdot p_{u,v}$
end function

$s_j(x) = \text{INTERPOLATOR}(y, k, 0)$

4.5 Mapping

An element from $\mathbb{F}_{p^2}$ is composed of a pair of elements from $\mathbb{F}_p$. Thus, each m bits of source data can be mapped into an $\mathbb{F}_p$ element as long as 0 and $2^m - 1$ are not both part of the source data, given that they cannot be distinguished ($2^m - 1 \equiv 0 \mod p$). The following paragraph describes one possible approach for such case.

For a content s represented by n elements from $\mathbb{F}_{p^2}$ (of size $2 \cdot m \cdot n$ bits), where $n < p/2$, there are, at most, $p - 2$ unique elements from $\mathbb{F}_p$ $[2 \cdot (p/2 - 1)]$. Thus, there is at least one element a from $\mathbb{F}_p$ that is not in the set of $2 \cdot n$ elements: $\forall s \in \mathbb{F}_p^{2 \cdot n},\ \exists a \in \mathbb{F}_p : a \notin s$. This element can be used to replace $2^m - 1$ in the content data whenever it occurs, before encoding, and do the reverse after decoding. For contents too large to be represented this way ($n \geq p/2$), the content data can be treated the same way by dividing it into several partitions of, at most, $p/2 - 1$ elements from $\mathbb{F}_{p^2}$. This information can be included, e.g., in a torrent file, and only adds, at most, m bits per partition. Over the finite field $\mathbb{F}_{(2^{31}-1)^2}$, this represents only 31 additional bits for contents of a size up to approximately 7.75 Gibibytes (GiB).

4.6 Encoding

Section 4.1 defines the encoding process as the transformation $\mathbf{C} \xrightarrow{\mathcal{T}} \mathbf{B}$, which is obtained by applying the transformation $(s_{0,j}, \ldots, s_{k-1,j}) \xrightarrow{\mathcal{F}} (e_{0,j}, \ldots, e_{n-1,j})$ over $\mathbb{F}_q^n$ to each one of the d columns of matrix $\mathbf{C}$, $0 \leq j < d$. Extending the source vector s_j with $n - k$ zeros to make it of size n, $s_j = (s_{0,j}, s_{1,j}, \ldots, s_{k-1,j}, 0, \ldots, 0)$, enables to generate n symbols, at once, using a size n CMNT. However, this approach does not enable n to increase efficiently as needed because all symbols have to be generated each time n is increased, including the ones generated previously.

In order to become rateless, Storm erasure codes decompose the transformation $\mathcal{F}$ into n/k steps to enable the generation of new blocks as needed, in groups of k. The

transformation $\mathcal{F}$ is decomposed taking into consideration that the output symbols of a CMNT are in bit-reversed order to ensure that the output of performing n/k CMNTs of size k is exactly the same as performing a single CMNT of size n with the source vector extended with $n-k$ zeros. Let ψ be an index, and Ψ the corresponding index with the bits in reverse order, then $e_{\psi,j} = S_{\Psi,j}$, i.e., the evaluation point x_ψ is the unity root W_n^Ψ. Expressing index Ψ as $\alpha \cdot n/k + \gamma$ enables to separate the relative index α of a size k CMNT from the index γ of the step or group of k symbols, where α represents the $\log_2(k)$ most significant bits of Ψ, $0 \leq \alpha < k$, and γ represents the $\log_2(n/k)$ less significant bits of Ψ, $0 \leq \gamma < n/k$. The expression to compute $S_{\Psi,j}$ after rewriting it as $S_{\alpha\cdot\frac{n}{k}+\gamma,j}$ is given by Equation 4.7.

$$S_{\Psi,j} = S_{\alpha\cdot\frac{n}{k}+\gamma,j} = \sum_{\beta=0}^{k-1} s_{\beta,j} \cdot W_n^{\left(\alpha\cdot\frac{n}{k}+\gamma\right)\cdot\beta} = \sum_{\beta=0}^{k-1} \left(s_{\beta,j} \cdot W_n^{\gamma\cdot\beta}\right) \cdot W_k^{\alpha\cdot\beta} \tag{4.7}$$

Let the source vector of each individual CMNT of size k be $s'_{\beta,j} = s_{\beta,j} \cdot W_n^{\gamma\cdot\beta}$, which corresponds to a frequency shift, the output of a size n CMNT can be obtained through n/k CMNTs of size k. If n increases to $2n$, the evaluation point x_ψ becomes $W_{2\cdot n}^{2\cdot\Psi}$, which is still the unity root W_n^Ψ but expressed as a $2n$-th root of unity, and thus previously generated symbols do not need to be recomputed. The encoding time complexity is then $\Theta\left(n/k \cdot k \log k = n \log k\right)$, given that applying the $W_n^{\gamma\cdot\beta}$ factors has a linear cost and therefore the impact on the overall time complexity is negligible. The described encoding algorithm creates a non-systematic erasure code.

A systematic construction can be obtained by applying an ICMNT to the source vector. This way, the first group of k symbols becomes the original source vector (the CMNT reverts the ICMNT), and all other groups of k symbols provide regular encoded symbols.

4.7 Decoding

The decoding algorithm works almost the same way for systematic and non-systematic constructions of Storm erasure codes, and only two minor differences set them apart: the decoding algorithm is only run for the systematic construction if the input does not contain all k data symbols; the systematic construction requires an additional CMNT to retrieve the data symbols. It consists in five main steps: 1) calculate the Lagrange basis polynomial, $L(x)$; 2) compute its derivative, $L'(x)$; 3) evaluate the barycentric weights as $\omega_\alpha = (L'(x_\alpha))^{-1}$; 4) compute all $y_{\alpha,j} = e_{\alpha,j} \cdot \omega_\alpha$; 5) perform the interpolation. Given that any set of k evaluation points is a subset of η-th roots of unity, the interpolation can be performed either at k arbitrary points or at η unity roots depending on the values of k and η, where η is the cardinality of the smallest set of unity roots that contains all k points. The interpolation at k arbitrary points was defined in Section 4.4, the interpolation at η unity roots is defined as follows.

Let $W_\eta^{z_\alpha}$ be the power representation of x_α, $0 \leq \alpha < k$, $0 \leq z_\alpha < \eta$, and $Y_j(x) = \sum_{\alpha=0}^{k-1} y_{\alpha,j} \cdot x^{z_\alpha}$, where $y_{\alpha,j} = e_{\alpha,j} \cdot \omega_\alpha$ as for the multi-point interpolation algorithm. Using the Taylor series of ${}^1/_{(x - W_\eta^{z_\alpha})} \mod x^\eta = -\sum_{\beta=0}^{\eta-1} W_\eta^{z_\alpha \cdot (-\beta-1)} \cdot x^\beta$, Lagrange's interpolation formula from Equation 4.2 can be rewritten as given by Equation 4.8 [57].

$$s_j(x) = -L(x) \cdot \sum_{\alpha=0}^{k-1} \left(y_{\alpha,j} \cdot \sum_{\beta=0}^{\eta-1} \left(W_\eta^{-\beta-1} \right)^{z_\alpha} \cdot x^\beta \right) = -L(x) \cdot \sum_{\beta=0}^{\eta-1} Y_j \left(W_\eta^{-\beta-1} \right) \cdot x^\beta \tag{4.8}$$

Let $N(k)$ be the time complexity of computing a CMNT/ICMNT of size k, $M(k)$ the time complexity of computing the product of two polynomials of degree $k-1$: assuming $N(2k) \approx 2 \cdot N(k)$, $M(k) \approx 6 \cdot N(k)$. Then, step 1) takes $M(k) \log k$ time from performing $\log k$ stages of polynomial products with time complexity equivalent to the product of two degree $k-1$ polynomials. Step 2) takes $\Theta(k)$ time to compute the derivative. Step 3) takes either $N(\eta)$ time at η roots of unity – $1 \cdot N(\eta)$ – or $M(k) \log k$ time at k arbitrary points – the multi-point evaluation algorithm performs $\log k$ stages, each in approximately $M(k)$ time. Step 4) also takes $\Theta(k)$ time. Step 5), using Equation 4.8, takes $M(\eta)$ time at η unity roots because evaluating $\sum_{\beta=0}^{\eta-1} Y_j \left(W_\eta^{-\beta-1} \right) \cdot x^\beta$ takes $N(\eta)$ time, and multiplying the result by $L(x)$ is performed in $M(\eta)$ time: $N(\eta) + M(\eta) \approx 7 \cdot N(\eta)$. At k arbitrary points, step 5) takes $M(k) \log k$ time. Therefore, step 5) has $\min\{M(\eta), M(k) \log k\}$ time complexity. The overall time complexity is $M(k) \log k + \min\{M(\eta), M(k) \log k\}$. However, in practice, the overall time complexity is just $\min\{M(\eta), M(k) \log k\}$ because steps 1) to 3) are only performed once as they only depend on x_α, while steps 4) and 5) are performed thousands of times to retrieve a content as chunks are composed of thousands of symbols. $L(x)$ also depends only on x_α and just needs to be computed once per content, thus step 5) can be performed in $5 \cdot N(\eta)$ time at η roots of unity. As so, the decoding has $\min\{M(\eta), M(k) \log k\}$ practical time complexity: $\min\{5 \cdot N(\eta), 6 \cdot N(k) \cdot \log k\}$.

4.8 Performance Evaluation

Complexity analysis provides an understanding of how an algorithm behaves as the input grows, but it hides constant factors that may alter significantly the algorithm's practical performance. To assess the performance of Storm erasure codes, and to compare them with Soro et al.'s [57] – the only ones with $\Theta(n \log n)$ time complexity that admit any power of two for n and k, $k \leq n$ – both were implemented in C++ and ran on an Intel Core i5-560M under Ubuntu 13.10 64 bits. For evaluation, the Fermat field $\mathbb{F}_{2^{16}+1}$ and the complex Mersenne field $\mathbb{F}_{(2^{31}-1)^2}$ were used. The results depicted in Figure 4.1 are for a single thread.

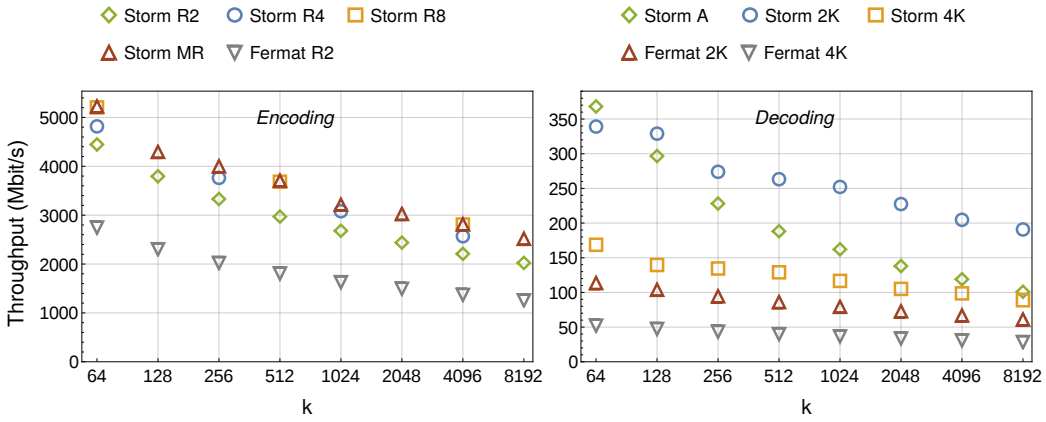


Figure 4.1: *Encoding and decoding throughput comparison.* Encoding throughput for radix-2, radix-4, radix-8, and mixed-radix over complex Mersenne finite field, and for radix-2 over Fermat field [left]. Decoding throughput using interpolation at k arbitrary points (*Storm A*), and at n roots of unity, with $n = 2k$ and $n = 4k$, over the same finite fields [right].

As expected, the performance improvement provided by radix-8 CMNT in comparison to radix-2 CMNT is significant – more than 25% for $k \geq 512$. The mixed-radix CMNT over $\mathbb{F}_{p^2}$, which uses higher radixes whenever possible, nearly doubles the throughput provided by radix-2 FNT over Fermat fields. When comparing only radix-2 NTTs, the larger symbols of $\mathbb{F}_{p^2}$ (62 vs 16 bits) improve performance despite multiplications being slightly more expensive (four integer multiplications and two additions). Identical results were obtained for decoding: the throughput for $n = 2k$ over complex Mersenne field, which is about twice the throughput for $n = 4k$ over that field, is slightly greater than twice the throughput for $n = 2k$ over $\mathbb{F}_{2^{16}+1}$. The decoding algorithm at k arbitrary points is more advantageous for small values of k and, for k up to 8192 when $n/k > 2$.

4.9 Conclusions

Storm erasure codes are rateless MDS erasure codes based on RS codes with $\Theta(n \log k)$ encoding time complexity and $\min\{M(\eta), M(k) \log k\}$ upper bound for decoding time complexity, and their practical performance was assessed and compared against Soro et al.'s [57]. Storm erasure codes are able to saturate a Gigabit interface on a CPU released in Q3 2010, and are able to provide nearly twice the throughput of equivalent codes defined over Fermat fields. Unlike Fermat fields, there is no known field size limit for $\mathbb{F}_{p^2}$.

Current implementation is single-threaded, and thus the intent is to create a parallel multi-core implementation. The development of a new interpolation algorithm that enables to combine both described polynomial interpolation algorithms – CMNT at unity roots and multi-point polynomial interpolation at arbitrary points – is under consideration. E.g., if the set of k evaluation points is composed of a set of $k/2$ unity roots and $k/2$ arbitrary points, one half could be interpolated at $k/2$ unity roots.

5 CIDRARCHY NS-3 ROUTING PROTOCOL

> *At the bottom, the elimination of spyware and the preservation of privacy for the consumer are critical goals if the Internet is to remain safe and reliable and credible.*
>
> **— Cliff Stearns**

Simulation plays a vital role on designing, building, understanding, and thoroughly evaluating large-scale Internet systems, which, in particular those based on P2P architecture such as BitTorrent, usually involve thousands or even millions of peers. For systems on such a scale, with heterogeneous peers (e.g. heterogeneous Internet access, CPU, memory, and storage resources), it becomes impracticable to test accurately the designed protocols either analytically or using real large-scale implementations. Small scale tests, although feasible, may not be enough as some issues may only arise, or some features may only be evaluated, at the scale of thousands of peers or more [10]. Simulations are only as good as their models, and the simulation of large-scale networks using accurate and complex models is a complex task. When simulating large-scale Internet systems, two important models are required: the Internet topology, and the network stack and its protocols. Simulators are usually forced to trade off simulation accuracy for scale [20] because it is hard to evaluate Internet systems over a large and complex Internet topology while using a complex and realistic network stack, and its protocols.

An Internet topology model is a conceptual two-level hierarchical network that connects hosts, routers, and Autonomous Systems (ASes) to each other in a way that mimics the Internet topology. Given that the simulation of Internet systems typically does not consider inter-domain (AS-level) topology and its routing, the target is the intra-domain topology and its routing. The reader is referred to [28] for further reading regarding AS-level topology. The work presented in this chapter follows the model provided in [3] that depicts a conceptual intra-domain network topology closely related to a real communications network of an ISP. Figure 5.1 illustrates such model, and depicts the hierarchical structure of such communication networks, which typically consist in access, aggregation, and core sections. The network termination represents hosts, which may connect using different technologies.

ns-3 [@4] is a free and open-source discrete-event network simulator that is mainly targeted for research and educational use. It is the successor of ns-2, the most popular network simulator for research [34], but ns-3 is a replacement rather than an extension of ns-2 as its core was rewritten to improve upon ns-2 limitations.

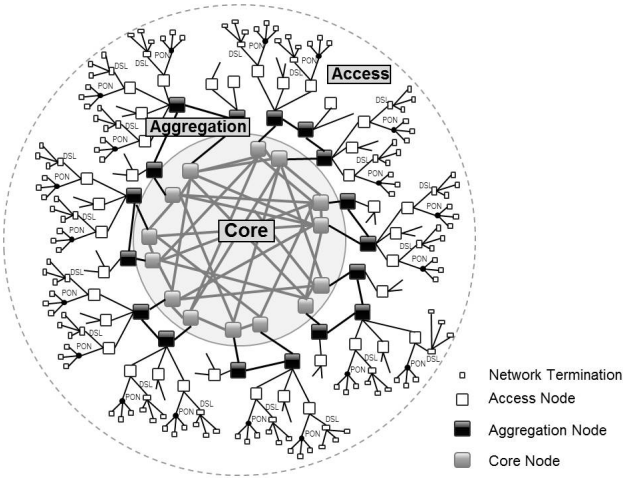


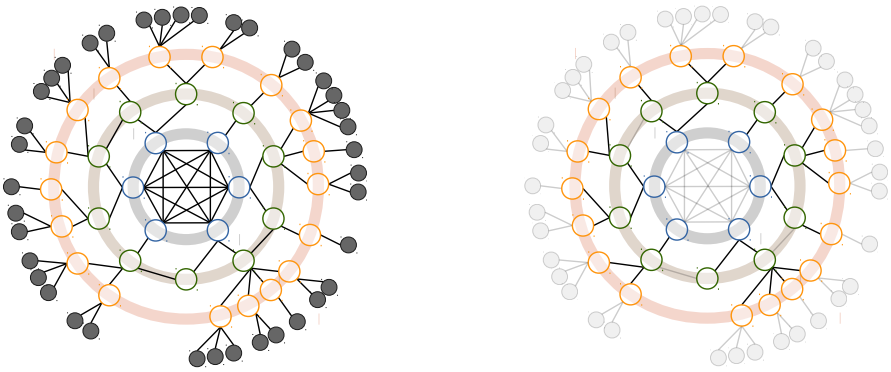
Figure 5.1: *Intra-domain network topology model.* Hierarchical structure of an intra-domain network, which is typically divided into access, aggregation, and core sections. The network termination represents hosts (figure obtained from [3]).

ns-3 simulates realistically the network stack but the scale and complexity of the Internet topology is limited by the IP forwarding of existing ns-3 routing protocol implementations. Several solutions were proposed to improve ns-3 performance, e.g., resorting on parallel [61] and distributed [43] computing, and avoiding redundant computations [25], but the IP forwarding remains an open issue for large-scale networks. ns-3 provides several routing protocols both for infra-structured and ad-hoc networks, the latter being the main subject of research in this area. The two main ns-3 routing protocols for infra-structured networks, `Ipv4GlobalRouting` and `Ipv4NixVectorRouting`, perform IP forwarding table lookup in linear and logarithmic time, respectively. Additionally, ns-3 does not provide a straightforward way to construct a topology with asymmetric links, as Internet access usually is.

The remainder of this chapter is structured as follows. Section 5.1 provides an overview of CIDRarchy, an ns-3 IPv4 routing protocol that performs IP packet forwarding in constant time to enable large-scale P2P network simulation. Section 5.2 depicts the implementation of an ns-3 helper to ease Internet-like network topology creation. Section 5.3 describes the implementation of the ns-3 IPv4 routing protocol itself and of its helper, along with automatic IPv4 address assignment. On Section 5.4 a performance comparison between CIDRarchy, `Ipv4GlobalRouting` and `Ipv4NixVectorRouting` is provided. Section 5.5 draws the main conclusions.

5.1 Overview

CIDRarchy is an ns-3 IPv4 routing protocol that uses CIDR[21] as the base to create a hierarchical Internet-like network topology, hence its name (CIDR + hierarchy), in order to enable automatic IPv4 address assignment and IP forwarding with constant time complexity, i.e., that does not depend on the IP forwarding table size. Albeit being out of the scope of this work, the same rationale can be



(a) Network topology with all links. **(b)**Subnetwork management hierarchy.

Figure 5.2: *Example of an asymmetric network topology supported by CIDRarchy and the hierarchy used to set subnetwork management.* An example topology created using `AsymmetricHierarchicalTopologyHelper`, where routers are represented as empty circles, and hosts are represented as filled circles [left]. Hierarchy used to set the subnetwork managed by each router (represented as empty circles) [right].

used to extend CIDRarchy in order to support IPv6[1]. CIDRarchy provides an ns-3 helper (`AsymmetricHierarchicalTopologyHelper`) to ease network topology creation with asymmetric links, and an ns-3 helper (`Ipv4CidrarchyHelper`) to install the ns-3 routing protocol (`Ipv4CidrarchyRouting`), and to automatically assign IPv4 addresses. For the sake of clarity, given that classes have self-descriptive names and their suffixes indicate their purpose – e.g., a helper, a channel or a net device –, only the class name is used. Its plural may also be used, although always referring to the same class.

All links between nodes are created using a `PointToPointChannel`, so that uplink and downlink bitrates, delay, and Maximum Transmission Unit (MTU) can be configured per link, following the referred intra-domain network topology model. The `AsymmetricHierarchicalTopologyHelper` enforces a node hierarchy per levels, allowing as many levels as required, but in which links can only be created between nodes at the same level or at adjacent levels. It is assumed that core routers form a fully connected mesh network so that packets are always forwarded through the minimum hop count route, and that hosts access the network through a single direct link to an access router and do not establish any other direct link with any other network node. Routers connect either hosts (access routers) or other routers (aggregation and core routers), but there is only one direct link between any pair of routers. Also, they have a single link to a router on the level above (parent router), except for core routers (first level) which have none. Figure 5.2a depicts an example of a network topology that can be constructed using `AsymmetricHierarchicalTopologyHelper`.

CIDRarchy ns-3 routing protocol requires a strict hierarchical assignment of the subnetworks managed by each router to achieve constant time complexity: the subnetworks managed by sibling routers have equal length prefix, and are aggregated

[1] Internet Protocol version 6.

in a single CIDR prefix at parent router. Thus, e.g., a router having three child routers, managing the subnetwork `27.2.13.0/24`, would assign the subnetworks `27.2.13.0/26`, `27.2.13.64/26`, and `27.2.13.128/26` to its child routers. This procedure is applied recursively from top to bottom, first by assigning equal CIDR prefixes to each core router, and then by treating each subnetwork as a tree topology: ignoring the links between routers at the same level, each subnetwork becomes a tree. Figure 5.2b illustrates this approach, showing the hierarchy used to set the subnetwork managed by each router. Access routers sequentially assign IPv4 addresses to hosts within the subnetwork they manage.

The `Ipv4CidrarchyHelper` provides an `Install` method that receives, besides the network prefix, three node lists as input to perform all network setup operations: core routers, routers, and hosts lists. The list of core routers is required to bootstrap the assignment of the subnetwork managed by each router. The list of routers is required to complete network access setup. The list of hosts is used to automatically assign IPv4 addresses to them. CIDRarchy supports the addition of hosts in runtime as long as there are IPv4 addresses available at the intended access router.

5.2 ns-3 Helper for Topology Creation

This section describes the `AsymmetricHierarchicalTopologyHelper`, which enables the creation of topologies akin to the model of ISP networks provided in [3], and provides code samples in C++ to depict its usage. It is shown how the access network is created, explained how the hosts connect to it, and depicted how star, tree (balanced and unbalanced), and mesh network topologies can be created using `AsymmetricHierarchicalTopologyHelper`. These typical network topologies are then used to compare CIDRarchy against existing ns-3 routing protocols.

The `AsymmetricHierarchicalTopologyHelper` constructs the network topology following a top-down approach, i.e., the network links are created by providing either the router on the level above (parent router) or a router at the same level (for access or aggregation routers). The network nodes are connected using asymmetric `PointToPointChannels` (`AsymmetricPointToPointLinks`), configurable full-duplex links between a pair of `PointToPointNetDevices`. The `PointToPointHelper` provides the means to easily create and configure several properties of a symmetric `PointToPointChannel`, such as delay, MTU, and bitrate, but does not enable asymmetric uplink and downlink bitrates. Therefore, the `AsymmetricPointToPointLink` class provides an interface to enable, among other functionalities, straightforward and independent configuration of downlink and uplink bitrates. In order to avoid unnecessary verbosity and error-prone repetition, default link properties can be configured both for host links (between a host and an access router), and for router links (between a pair of routers, including core routers) when using the `AsymmetricHierarchicalTopologyHelper`. This helper also enables to create multiple links at once, both host and router links, and in which case the same configu-

ration is applied to all links: either the default configuration or the one provided through parameters.

Depending on the required level of accuracy, the ISP network can be instantiated as a star network, a tree network or a mesh network. Star network topology can be created using the `AsymmetricHierarchicalTopologyHelper` by defining a single router to which all hosts connect to, i.e., there is a single level of routers with only one router. A tree topology can be obtained by creating one router at the first level, and adding additional levels with routers as required. Finally, a mesh topology can be obtained either implicitly, by creating several routers at the first level, or explicitly, by creating links between routers at the same level. The following paragraphs describe each one of these network topologies.

Listing 5.1 provides a code sample for the creation of a star topology. The helper is declared on line 1, and the default configurations for router and host links are provided on lines 4 and 5, respectively. On line 7, a new router with no parent is created (`nullptr` is a C++11 keyword for the null pointer), i.e., a root router. Then, 1000 host links are added to this router using their default configuration on line 8.

Listing 5.1: Star Topology Creation.

```
1 AsymmetricHierarchicalTopologyHelper helper;
2
3 // downlink, uplink, link delay, and MTU
4 helper.SetRouterLinkDefaults("100Gbps", "100Gbps", "5us", 1500);
5 helper.SetHostLinkDefaults("30Mbps", "3Mbps", "5ms", 1500);
6
7 Ptr<Node> root = helper.CreateRouter(nullptr);
8 helper.CreateNHosts(1000, root);
```

The creation of a balanced tree is depicted by Listing 5.2, where the helper creation and the default configuration for host and router links is omitted. Line 1 creates a root router to which the routers declared on lines 2 to 4 connect to. The *left* router uses the default configuration, and the *right* router is configured with 100 Gbit/s downlink, 100 Gbit/s uplink, 5 microseconds delay, and 1500 bytes MTU. On lines 6 to 9, 500 hosts are linked to each access router (*left* and *right*), using 30 Mbit/s downlink, 3 Mbit/s uplink, 5 miliseconds delay, and 1500 bytes MTU links.

Listing 5.2: Balanced Tree Topology Creation.

```
1 Ptr<Node> root = helper.CreateRouter(nullptr);
2 Ptr<Node> left = helper.CreateRouter(root);
3 Ptr<Node> right =
4   helper.CreateRouter(root, "100Gbps", "100Gbps", "5us", 1500);
5
6 NodeContainer leftHosts =
7   helper.CreateNHosts(500, left, "30Mbps", "3Mbps", "5ms", 1500);
8 NodeContainer rightHosts =
9   helper.CreateNHosts(500, right, "30Mbps", "3Mbps", "5ms", 1500);
```

The creation of an unbalanced tree topology is akin to the creation of a balanced counterpart. Therefore, Listing 5.3 depicts the creation of the unbalanced tree topology defining the *left* and *right* routers in an alternative way: using `CreateNRouters` function (lines 2 to 4); the signature of `CreateN` functions has, additionally, the number of nodes to be created as first parameter. Line 1 creates the root router, and line 6 creates 998 hosts connected to the *left* router (index 0 on the `NodeContainer`). Line 7 declares a `NodeContainer` for right hosts, and lines 9 to 11 add one host to it using the configuration provided. On line 12, another host is added but using the default host link configuration. In this example, the unbalanced tree topology is thus composed of 998 hosts on the left branch, and 2 hosts on the right branch.

Listing 5.3: Unbalanced Tree Topology Creation.

```
1  Ptr<Node> root = helper.CreateRouter(nullptr);
2  NodeContainer access = helper.CreateNRouters(
3    2, root, "100Gbps", "100Gbps", "5us", 1500
4  );
5
6  NodeContainer leftHosts = helper.CreateNHosts(998, access.Get(0));
7  NodeContainer rightHosts;
8
9  rightHosts.Add(
10   helper.CreateHost(access.Get(1), "30Mbps", "3Mbps", "5ms", 1500)
11 );
12 rightHosts.Add( helper.CreateHost(access.Get(1)) );
```

Lastly, Listing 5.4 provides a code sample for the creation of a mesh topology with two levels of routers: core and access sections. The `CreateNRouters` function is used to create 5 core routers (with no parent), and line 2 declares a `NodeContainer` for all access routers. The `for` loop on lines 4 and 5 creates 10 access routers, 2 routers per core router (arborescence of two). Lines 7 and 8 add 100 hosts to the network, 10 hosts to each access router (arborescence of ten). The `CreateSameLevelRouterLink` function could have been used to create links between routers at the same level (the first two parameters are the pointers to the nodes), and to which it is applied either the default configuration, if no additional parameters are provided, or the configuration provided when invoking the function.

Listing 5.4: Mesh Topology Creation.

```
1  NodeContainer core = helper.CreateNRouters(5, nullptr);
2  NodeContainer access;
3
4  for(uint32_t i = 0; i < core.GetN(); ++i)
5   access.Add(helper.CreateNRouters(2,core.Get(i)));
6
7  for(uint32_t i = 0; i < access.GetN(); ++i)
8   helper.CreateNHosts(10, access.Get(i));
9
10 // helper.CreateSameLevelRouterLink(access.Get(1), access.Get(5));
```

5.3 ns-3 IPv4 Routing Protocol

This section presents the CIDRarchy ns-3 routing protocol, a CIDR-based ns-3 IPv4 routing protocol with constant time complexity, and the ns-3 helper used to install it and to add new hosts during simulation runtime (`Ipv4CidrarchyHelper`). It describes how routers manage CIDR prefixes and assign IPv4 addresses to hosts, details how the forwarding decisions are performed, and discusses the implementation of the ns-3 routing protocol and its helper in ns-3 network simulator. Nodes that have the same parent router are referred as its children, and as siblings of each other.

ISP networks are hierarchically structured, static, and its routers are neither expected to generate nor to receive packets. Existing ns-3 routing protocols are general purpose and aim at supporting a wide range of network topologies, thus they cannot take advantage of the hierarchical and static structure of such intra-domain networks, and also assume that every node may generate or receive packets. On the contrary, CIDRarchy targets Internet systems, and is able to forward IP packets in constant time by taking advantage of the strict hierarchical assignment of the subnetworks managed by each router and by considering that only hosts can be the source or recipients of IP packets. The subnetwork managed by the parent router aggregates in a single CIDR prefix all the subnetworks managed by each child router, which are a result of dividing the parent subnetwork into child subnetworks with equal length prefix. Thereby, the number of child subnetworks is always a power of two. If multiple router levels are required, the subnetwork assignment is performed recursively in a top-down approach.

The IP packet forwarding decisions taken by the CIDRarchy ns-3 routing protocol can be seen as CIDR prefix adjustments to select an adjacent router that is one hop closer to destination host: the prefix widens (parent link) until the destination host is part of the subnetwork managed by the router, and it narrows once it does (child link). Except for core routers, which have no parent link, links to routers at the same level (peer links) are a shortcut to avoid going up and down the hierarchy. This enables CIDRarchy ns-3 routing protocol to decide how to handle an incoming packet at a router using, at most, four sequential constant-time checks. The packet is dropped once the recipient is found unreachable (inexistent forwarding link), forwarded downwards (child link) if a router that manages a subnetwork including the recipient's address has been found, forwarded sideways (peer link) if there is a link to a router at the same level that manages it, or forwarded upwards (parent link) if the subnetwork managed by the router is yet too small. The forwarding decision process is depicted by Figure 5.3 using a flowchart.

The forwarding decision process starts when an incoming packet arrives at a router, and the decision is made according to the IPv4 address of the intended recipient (*addr*). The first check is used to drop packets whose recipient is not within the network. The following two checks are used to determine the forwarding link, if any. If the router manages the subnetwork, then it forwards the packet through

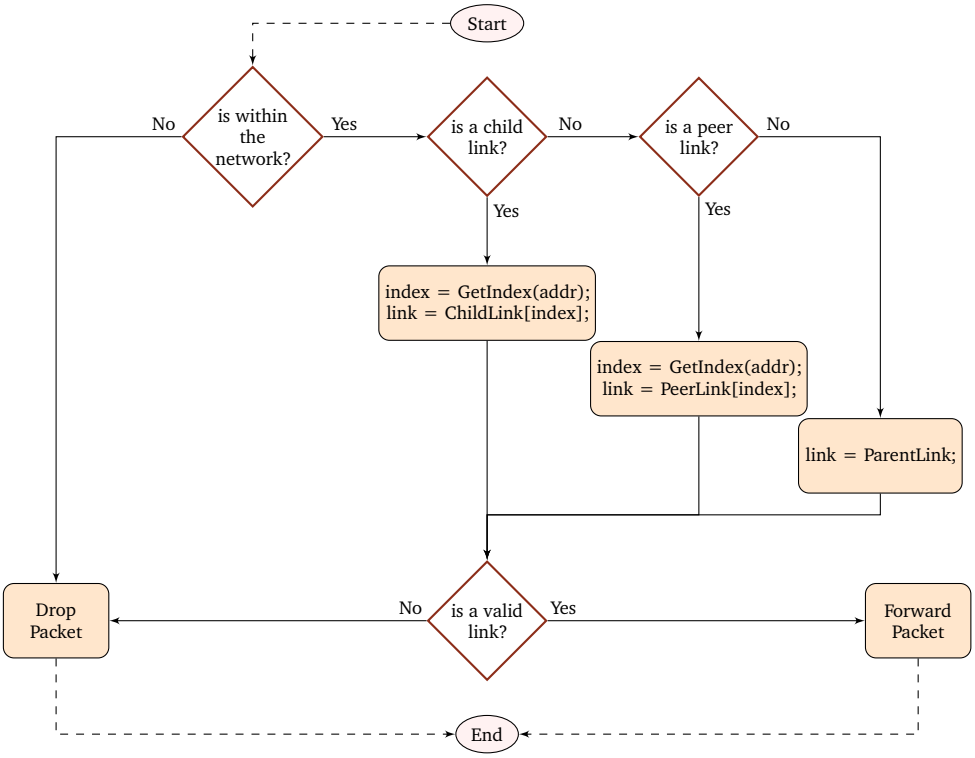


Figure 5.3: *IP packet forwarding decisions made by routers running CIDRarchy.* When a router receives an incoming packet with address *addr*, it performs, at most, four constant-time checks before deciding either to drop the packet or to forward it. If the forwarding link is either a child link or a peer link, the theoretical function `GetIndex` provides its index, also in constant time as the subnetworks have equal prefix length.

one of its child links; otherwise, it checks if there is a peer router that manages that subnetwork, and, if so, it forwards the packet through that link. If neither the router nor an eventual peer router manage the subnetwork, the packet is forwarded through the parent link to increase the size of the subnetwork currently managed. The final check is used to verify if the recipient is still potentially reachable, i.e., if there is a valid link to explore. There may be no such link in the following cases. Core routers, as well as sibling routers, have equal length prefix, and thereby all IPv4 address ranges are managed if, and only if, the number of core routers or sibling routers at each level is a power of two. As so, when a core router attempts to forward a packet whose recipient is within an unmanaged subnetwork, the selected forwarding link is necessarily invalid; if a router manages the subnetwork but there is no child router managing the subnetwork within, the link is also necessarily invalid. An access router may also select an invalid child link if the IPv4 address of the recipient has not yet been assigned. The function represented as `GetIndex` runs also in constant-time: the index of the link is given by the bits that are part of child's subnetwork prefix but not of parent router's subnetwork prefix. For instance, a router managing the subnetwork `27.2.13.0/24` that needs to forward a packet to host `27.2.13.66`, will select the child link with index 1 (the router managing subnetwork `27.2.13.64/26`). The same rationale is applied to routers at the same level, selecting the longest subnetwork prefix as the base (smallest subnetwork), and handling subnetworks with shorter

prefix (larger subnetwork) as multiple subnetworks of equal size. `ChildLink` and `PeerLink` are the arrays storing, respectively, the pointers to child and peer links; `ParentLink` is a pointer to the parent link.

CIDRarchy ns-3 routing protocol is implemented in the `Ipv4CidrarchyRouting` class without deriving it from `Ipv4RoutingProtocol` class, the base class for implementing ns-3 IPv4 routing protocols, given that routers are neither expected to generate nor to receive packets, and that the access network is static. `Ipv4L3Protocol`, the class that implements the IP layer, introduces considerable overhead in the way it processes the routes (`Ipv4Route`) passed by `RouteInput` and `RouteOutput` methods, and in the way it checks if an IP packet is to be locally delivered (invoking `IsDestinationAddress` method). `Ipv4Route` object contains the egress `NetDevice` but not the `Ipv4Interface`. For this reason, `Ipv4L3Protocol` iterates over the node's list of `Ipv4Interfaces` to perform a reverse lookup, which introduces overhead that varies with the interface list size. Given that every node is assumed to be a potential host, when destination address differs from the one of ingress interface, by default, `IsDestinationAddress` iterates over the same list to check if there is a match with the destination address. In order to avoid such overhead, CIDRarchy is implemented through a callback registration at `PointToPointNetDevice`, using `SetReceiveCallback` method, and by snooping the IPv4 header to perform the IP packet forwarding.

The `Ipv4CidrarchyHelper` helper provides an `Install` method that expects, besides the network prefix, three node lists as input to perform all network setup operations: core routers, routers, and hosts lists. Hosts, non-core routers, and core routers are treated differently as they require different configurations. The list of core routers is required to bootstrap the assignment of the subnetwork managed by each router, which, if peer links are ignored, is a set of trees (one per core router). Core routers have no parent link, and the assignment of child subnetworks is performed recursively in a breadth-first fashion. The list of routers is required to complete network access setup, and it is used to process peer links once the assignment of the subnetworks managed by each router has been completed. The list of hosts enables to automatically assign IPv4 addresses to them, and also to add a static route to their routing tables. Considering that the network topology has been previously setup by the `AsymmetricHierarchicalTopologyHelper`, Listing 5.5 provides a code sample that depicts CIDRarchy ns-3 routing protocol install.

Listing 5.5: CIDRarchy ns-3 routing protocol install.

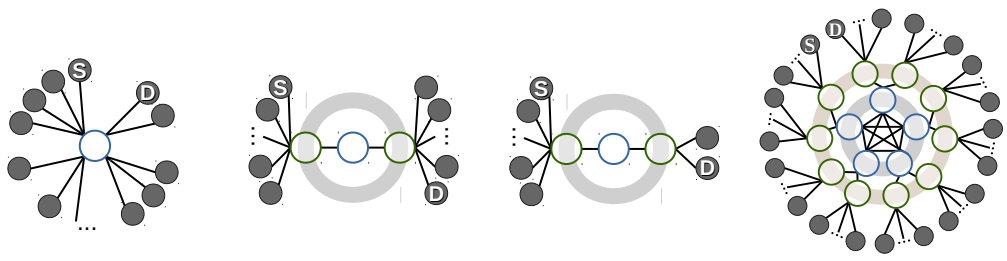
```
1 NodeContainer core = helper.GetCoreRouterNodes();
2 NodeContainer routers = helper.GetRouterNodes();
3 NodeContainer hosts = helper.GetHostNodes();
4 Ipv4CidrarchyHelper cidr;
5 Ipv4Address netwAddr = "10.0.0.0";
6
7 cidr.Install(core, routers, hosts, netwAddr, 16);
```

Lines 1 to 3 of Listing 5.5 show how the lists of core routers, routers, and hosts can be obtained from the `AsymmetricHierarchicalTopologyHelper` upon creating the desired network topology. Line 4 declares the `Ipv4CidrarchyHelper`, and line 5 declares the network address to be used. Then, on line 7, the `Install` method is invoked by providing, respectively, the list of core routers, the list of routers, the list of hosts, the network address (`10.0.0.0`), and the CIDR prefix length (`/16`). The `Install` method fails if the subnetwork managed by any router is empty, i.e., contains less than two assignable IPv4 addresses (`/31` prefix).

CIDRarchy supports the addition of hosts during simulation runtime, as long as there are IPv4 addresses available. Hosts can be added during simulation runtime using the class methods `CreateHost` and `CreateNHosts`, which take the same parameters as their counterparts of class `AsymmetricHierarchicalTopologyHelper`; there are no `CreateRouter` or `CreateNRouters` functions since the access network is considered to be static.

5.4 Results

This section compares CIDRarchy, `Ipv4GlobalRouting`, and `Ipv4NixVectorRouting` ns-3 routing protocols with respect to their performance and scalability, with the main objective of evaluating the impact of the IP packet forwarding on simulation time. For that purpose, first, it is measured the time required to simulate the transmission of single traffic flows on the four topologies previously described in Section 5.2: star, balanced tree, unbalanced tree, and mesh. Then, in a second study and in order estimate the overhead of ns-3 routing protocols bootstrap operations, it is used a balanced tree topology and measured the time required to simulate the transmission of multiple flows. The topologies used to conduct both studies are illustrated by Figure 5.4, and the traffic flows from host `S` (running `OnOffApplication`, an ns-3 built-in traffic generator that alternates between transmitting and idle states) to host `D` (running `PacketSink` application, an ns-3 built-in application to receive and consume that traffic). The results presented in this section are the average of five simulation runs on an Intel Xeon E5-2650@2GHz CPU, running on Ubuntu 14.04, using ns-3.22 version.



(a)star. **(b)**balanced tree. **(c)**unbalanced tree. **(d)**mesh.

Figure 5.4: *Topologies used for performance comparison.* The nodes `S` and `D` are the hosts running `OnOffApplication` and `PacketSink` applications, respectively.

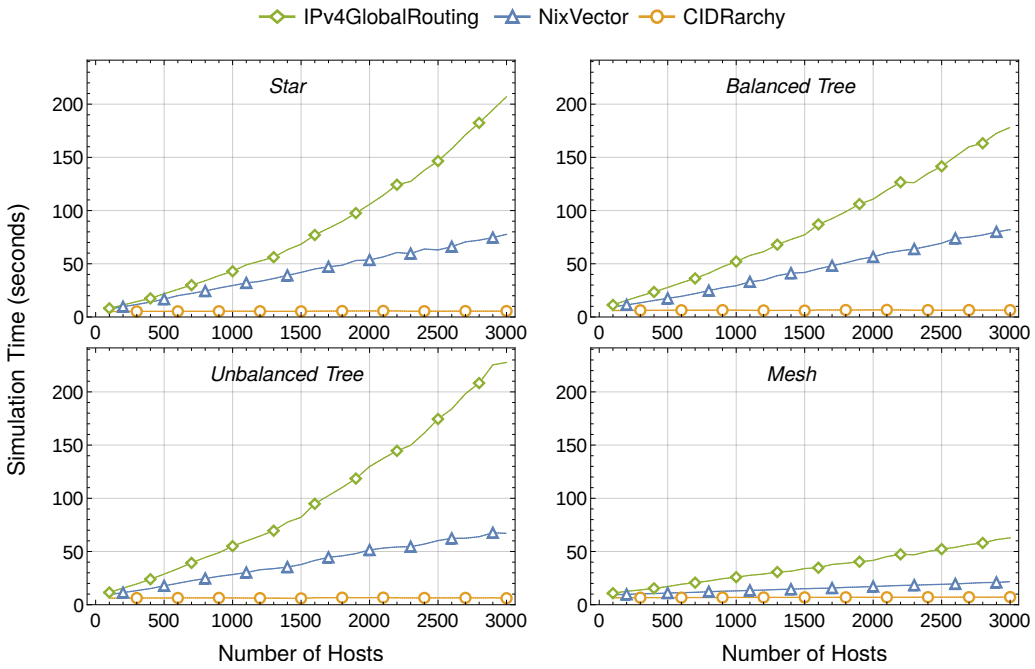


Figure 5.5: *IP forwading comparison between CIDRarchy, `Ipv4GlobalRouting` and `Ipv4NixVectorRouting`.* Simulation time required by CIDRarchy, `Ipv4GlobalRouting` and `Ipv4NixVectorRouting` ns-3 routing protocols to transmit 150000 packets, each packet having a size of 1000 bytes, using `OnOffApplication` and `PacketSink` applications, in star, balanced tree, unbalanced tree, and mesh scenarios with an increasing number of hosts.

Figure 5.5 shows the results of simulating the transmission of a single traffic flow consisting of 150000 packets, each packet having a size of 1000 bytes, for an increasing number of hosts. The hosts are connected as follows: *star* – all hosts connect the root router; *balanced tree* – half of the hosts connect to the *left* router and the other half connects to the *right* router; *unbalanced tree* – all but two hosts connect to the *left* router and two hosts connect to the *right* router; *mesh* – each access router connects the same amount of hosts (one-tenth). CIDRarchy presents similar simulation times as the number of hosts increases from 100 to 3000, and the slight increase can be explained by the additional computations required for network creation. On the other hand, despite the fact that the amount of packets exchanged remains the same, `Ipv4GlobalRouting` and `Ipv4NixVectorRouting` show a significant increase on simulation times as the number of hosts increases. CIDRarchy simulation time increases slightly as the hop count increases showing best results for the star scenario, which has a hop count of 2. `Ipv4GlobalRouting` and `Ipv4NixVectorRouting` simulation times decrease in the mesh scenario, which has hop count of 5. This behavior shows that the performance of both ns-3 routing protocols varies across different topologies, and also that IP forwarding has a significant impact on simulation time: simulation times of mesh scenario are three or four times lower than for other scenarios albeit having higher hop count. On all four scenarios, the simulation times of CIDRarchy ns-3 routing protocol are much lower than those of other ns-3 routing protocols, and the gains can reach over one order of magnitude.

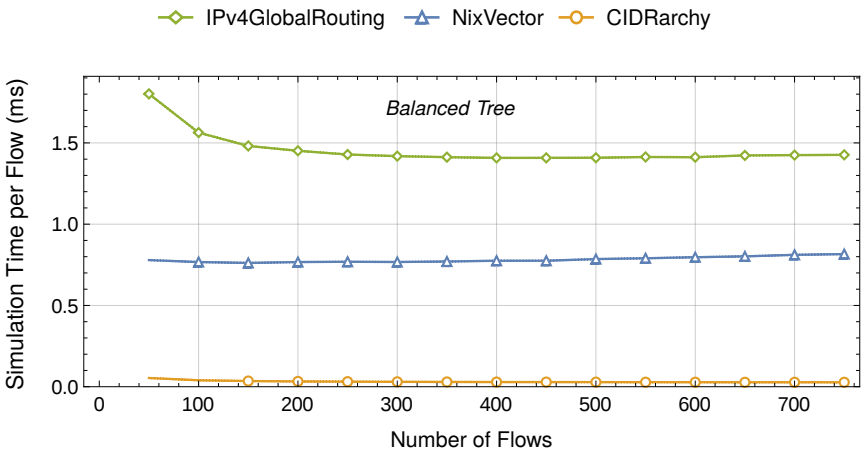


Figure 5.6: *Simulation time per flow required by CIDRarchy, `Ipv4GlobalRouting` and `Ipv4NixVectorRouting` ns-3 routing protocols.* Required simulation time per flow to transmit flows of 1500 packets, each having a size of 1000 bytes, using `OnOffApplication` (left branch) and `PacketSink` (right branch) applications, in a balanced tree topology with 1500 hosts in each branch (3000 hosts total), for an increasing number of flows.

Figure 5.6 shows the average simulation time, per flow, required by CIDRarchy, `Ipv4GlobalRouting` and `Ipv4NixVectorRouting` ns-3 routing protocols to transmit an increasing number of flows, each consisting of 1500 packets with a size of 1000 bytes, from the left branch hosts (running `OnOffApplication`) to the right branch hosts (running `PacketSink` application). The study was conducted using a balanced tree topology with 1500 hosts in each branch (3000 hosts total), and the average simulation time per flow is obtained by dividing the total simulation time by the number of transmitted flows. The average simulation time per flow remains nearly constant for both CIDRarchy and `Ipv4NixVectorRouting`, while it decreases for `Ipv4GlobalRouting` from 1.80 ms per flow (50 flows) until it stabilizes at around 1.43 ms per flow (250 flows). This shows that the initial cost of populating the `Ipv4GlobalRouting` routing tables is not negligible when considering short simulations. Despite considering only scenarios where each host communicates with a single host, which enables `Ipv4NixVectorRouting` to perform IP forwarding having only a single entry on its routing table and not to incur on any penalty that may arise from on-demand route calculation, CIDRarchy still outperforms it by at least one order of magnitude no matter the number of flows simulated.

5.5 Conclusions

CIDRarchy was implemented and evaluated in ns-3 simulator, and the simulation time gains over existing ns-3 routing protocols can reach over one order of magnitude. Thus, it enables to simulate large Internet networks in ns-3 by reducing greatly the time spent in IPv4 forwarding: e.g., a simulation taking one month to complete may require less than four days to complete if using CIDRarchy. It is provided also a

helper (`AsymmetricHierarchicalTopologyHelper`) that enables complex topology creation with a few lines of code, and new hosts can be added at runtime.

As future work, the following improvements are planned: enable variable length prefixes so that unmanaged networks can be avoided using any number of core or sibling routers; enable multiple host connections to support multi-homing scenarios; provide a packet drop callback that can be used, for example, to easily and efficiently integrate CIDRarchy with other lower priority ns-3 routing protocols.

6 Validation

> *Civilization is the progress toward a society of privacy. The savage's whole existence is public, ruled by the laws of his tribe. Civilization is the process of setting man free from men.*
>
> **— Ayn Rand**

The Mistrustful P2P model provides plausible deniability through *content interest disguise*. It has no trust requirements, prevents user legal liability in case of legitimate usage, and ensures deterministic protection of user content interests against attacks of a size up to a configured level. Users can configure the required trade-off between privacy and performance by setting the size c of the largest group of colluding peers to be protected against, and the minimum network overhead m of content interest disguise (minimum network disguise overhead).

The remainder of this chapter is structured as follows. Section 6.1 provides a security analysis of the Mistrustful P2P model to validate its privacy claims, assuming that, as described in Section 1.1 (Legal and Ethical Framework), a legitimate user should not be held legally liable as long as the main motivation to use a P2P system is to share legit contents, and it cannot be proven that the user had access to the content data. Section 6.2 presents the performance evaluation of the Mistrustful P2P model for a large number of experiments, and compares it against the one of a traditional P2P model, which provides no privacy protection.

6.1 Security Analysis

This section provides the security analysis of the Mistrustful P2P model. First, the common attacks in P2P systems are described, in the context of this work, along with the countermeasures employed. Then, the identification of user content interests, the proof of full content download, and user legal liability are discussed.

6.1.1 Common Attacks and Countermeasures

The most common attacks in P2P file sharing that are potentially applicable to the Mistrustful P2P model are those aiming mostly at compromising either privacy – Sybil, peer selection, and collusion attacks –, or the content sharing – forgery and repetition attacks, pollution attacks, and Denial-of-Service (DoS) or distributed DoS (DDoS). Table 6.1 summarizes these attacks along with the countermeasures

Table 6.1: *Summary of countermeasures employed to prevent common P2P attacks.*

Attack	Countermeasures
DoS	Blacklist misbehaving peers (at application level).
Sybil and Collusion	Avoid block advertisement; employ the disclosure constraint mechanism; treat multiple peers using related IP addresses as pseudonymous identities of a single entity.
Peer Selection	Use multiple unrelated trackers for the same content.
Forgery and Repetition	Peers communicate through direct links.
Pollution	Each block has a checksum signed by the publisher.

employed, focusing on the context of this work. The attack model considered, as described in Section 3.3 (Attack Model), assumes that an attacker can be any entity participating in the system – a publisher, a tracker, a seeder or a commoner – or a group of any of those entities in collusion. External entities, such as ISPs and governments, are out of the scope of this work.

DoS A denial-of-service attack aims either at shutting down the entire system or at making one or more contents unavailable. It may target trackers to prevent peers from knowing each other and therefore disable content sharing, or target seeders to prevent new blocks from being introduced to harden or even preclude content download.

At the application level, DoS attacks are prevented by blacklisting attackers; at the network level, the Mistrustful P2P model presents the same vulnerabilities as any other P2P file sharing system.

Sybil and Collusion The attacker creates a large number of pseudonymous identities (Sybil) or multiple malicious peers may coordinate their efforts (collusion) to increase the amount of blocks disclosed by other peers to either prove content download or determine user content interests. The size of the attack is defined by the number of unique peers (not known to be related).

As discussed in Section 3.3, both Sybil and collusion attacks of a size up to c are thwarted by not advertising what peers download or miss (avoiding block advertisement), and by employing the disclosure constraint mechanism. Avoiding block advertisement forces attackers to engage in the content sharing in order to gather information about the blocks owned by peers, increasing the resources required to launch such attacks. The disclosure constraint mechanism limits the amount of information that a single colluding group can gather. This protection may be extended by enabling the user to treat multiple peers using the same public IP address, such as those behind NAT, or related public IP addresses, such as those of a single organization, as pseudonymous identities of a single entity.

Peer Selection Trackers may narrow the advertised lists of peers to increase the amount of block requests sent to malicious peers and thereby increase the amount of blocks disclosed to them.

Peer selection attacks are avoided by registering at multiple unrelated trackers and requesting from them lists of peers in a swarm. As so, through comparison, misbehaving trackers can be identified and blacklisted. Nevertheless, this attack does not increase the amount of information that a single colluding group of a size up to c can gather.

Forgery and Repetition These attacks try to either tamper with the data being transmitted or to retransmit authentic data previously captured in order to achieve their goals.

We assume that attackers, being participants of the system, cannot forge the source IP address of packets (IP address spoofing). As so, they are not able to forge nor to repeat packets because peers communicate directly. Even considering IP address spoofing, as long as key exchange and distribution mechanisms are employed, which are out of the scope of this work, the integrity of packets can be ensured and the addition of a sequence number prevents repetition attacks.

Pollution Malicious peers may share polluted blocks to waste resources and decrease the overall performance by making compliant peers to further share them.

Pollution attacks are thwarted by employing asymmetric cryptography to sign checksums of each block in order to verify its integrity. The publisher may, e.g., distribute the public key and the checksums along with content description or metadata.

6.1.2 Determine User Content Interests

The Mistrustful P2P model provides deterministic protection of user content interests against passive attacks of any size, and against active attacks of a size up to c. Passive attacks are defeated by avoiding block advertisement. Active attacks are thwarted by constraining the amount of blocks implicitly disclosed to an attacker of a size up to c to be at most m, and by downloading at least m blocks per cover content. The provided protection is deterministic because contents are only downloaded if the privacy requirements are met. If the size of an attacker exceeds c, then the provided protection may become probabilistic: the attacker may be able to know if the peer fully downloaded a content (genuine), and thus know that the user has interest in that content. However, deterministic protection of user content interests is still provided against all other attackers of a size up to c.

The identification of genuine and cover contents may only be possible if the size of the actual attacker exceeds the size c of the largest colluding group considered (underestimated attacker). Still, the identification of a content as cover, per se, only enables an attacker to reduce the set of contents that the user may be interested in, given that the user may have no interest in any of them. Thereby, an underestimated attacker still has to prove content download in order to determine user content interests, as long as the actual amount of blocks downloaded per cover content can take any value between m (inclusive) and k (exclusive).

6.1.3 Prove Content Download

The protection provided by the Mistrustful P2P model is deterministic as long as the size of the actual attacker does not exceed c. Therefore, herein are discussed the necessary conditions for an underestimated attacker of size $c + \delta$ to be able to prove content download, where δ is the difference between the actual and the configured sizes of the largest attacker.

The disclosure constraint mechanism ensures that, at most, m blocks can be disclosed to any set of c peers. Let σ be the minimum amount of blocks disclosed to any of the top c peers, which can be, at most, m/c (the case in which the same amount of blocks is disclosed to all c peers). Then, given that the amount of blocks that can be disclosed to each one of the δ peers is at most σ, an underestimated attacker may be able to prove content download if, and only if, $m + \delta \cdot \sigma \geq k$. I.e., the provided protection is still deterministic for $0 < \delta < (k - m)/\sigma$.

For $\delta \geq (k - m)/\sigma$, the provided protection becomes probabilistic, as provided by other P2P privacy-preserving systems, and an underestimated attacker may be able to prove content download.

6.1.4 Legal Liability

The multitude and complexity of copyright laws across the globe make it impossible to define clear boundaries regarding user legal liability. Still, the user should not be held legally liable for, unknowingly and unwillingly, downloading an illegal content if the main motivation to use a P2P system is to share legit contents, and it cannot be proven that the user had access to the content data.

The extent of user liability and the strength of plausible deniability depend on the main motivation to use the system, and on the type of contents that are distributed by the P2P file sharing system: if the system is mostly used to distribute illegal content, it is less plausible that the user intended to share legit contents when using it. Therefore, it is important to endow the content interest disguise scheme with the means to distinguish legit from illegal contents in order to ensure that the main motivation to use the system comes mostly, if not completely, from legit content sharing.

A legitimate usage of a P2P system based on the Mistrustful P2P model may only result in the download of an illegal content either due to inadvertently considering an illegal cover content as legal or due to misleading description. For the first case, given that the user is never granted access to the content data of cover contents as they are never fully downloaded, the user is not subject to any legal liability. For the second case, as long as actual attacker is not underestimated, the user is also not subject to legal liability because it cannot be proven that the user had access to the content data. Even if a misleading content download can be proven by an underestimated attacker, as long as the percentage of such misleading contents

remains low, it is still plausible that the user may have been driven to unknowingly and unwillingly download that content. Thereby, even in such case, the user may be able to avoid being subject to any legal liability.

6.2 Performance Evaluation

The performance evaluation of the Mistrustful P2P model was conducted through simulation, and, given that simulations are only as good as their models, they were carried out using the ns-3 discrete-event network simulator, which provides realistic models of the network stack and its protocols. Still, the simulation of large-scale P2P networks using accurate models generates a very large number of events so the time required to run simulations is large.

In order to be able to simulate P2P content sharing with several thousands of peers using accurate network layer models, the CIDRarchy ns-3 routing protocol was used. CIDRarchy takes advantage of the hierarchical structure of Internet-like network topology to minimize the computational cost of performing IP packet forwarding without changing the accuracy of the models. Simulating the application layer also requires modeling, among other functions, peer arrivals and departures, and peer join-participate-depart cycles (sessions), as the content download may span across multiple sessions [60].

The two main goals of the performance evaluation are to show that peers are able to timely download contents without advertising what they download, and to estimate the impact of privacy preservation on the average download bitrate; optimizing the overall performance of the Mistrustful P2P model is out of the scope of this work. To do so, the content sharing was simulated in order to evaluate the ratio of peers that are able to complete their downloads and the average download bitrate. The results obtained were compared against those of a traditional P2P file sharing model. As referred in Section 1.3, a content is considered to have been timely downloaded if the average download bitrate is within the same order of magnitude of traditional P2P file sharing models.

The performance evaluation aims at assessing the impact of content size, peer arrival rate, number of seeders, c and m, and cover downloads on the average download bitrate and on the download completion ratio. For the sake of clarity and tractability, the aim is at reducing the impact of other variables on the overall performance, including those referred in the literature [24, 60] such as session length, peer lifetime (time between first arrival and last departure), downtime, uptime, lingering time (additional time a peer lingers in the system after download completion), and inter-content relations. It is considered that peers have homogeneous Internet connections, do not perform simultaneous chunk requests, always attempt to complete the download in a single session, and leave immediately after completing the download (worst case). Also, it is considered a single genuine con-

tent download in order to enable fair performance comparison with traditional P2P systems, and thus cover downloads are emulated by increasing the peer arrivals and by having those peers to download only a fraction of the content.

The remainder of this section is organized as follows. First, it is described how peer arrivals are generated from real peer arrival traces. Then, the simulation setup for both P2P models is characterized. Lastly, the experiments considered are defined.

6.2.1 Peer Arrivals

A content download is usually broken into three stages: *flash crowd*, *steady-state*, and *end phase* [7, 46]. The *flash crowd* is the most demanding stage because there is a sudden burst of peer arrivals, which largely surpass the peer departures. The *steady-state* stage is characterized by an equilibrium between arrivals and departures. The *end phase* stage comprehends the end of life of a content where there are fewer arrivals than departures. As so, an ordinary Poisson arrival process is not able to capture the peer arrival dynamics because the mean peer arrival rate changes over time.

In order to ensure that the peer arrival rates are realistic, they are obtained as follows. The peer arrival traces of several contents were gathered, and then an exponential function is used to generate the peer inter-arrival times with a mean peer arrival rate that changes every 10 minutes (non-homogeneous Poisson process). The traces were gathered by monitoring a widely used tracker (*open.demonii.com*), and provide the number of first time peer arrivals over 10 minute intervals since content publication up to 21 days. The gathered peer arrival traces were not used directly because trackers, per request, provide a list of, at most, 200 peers currently in a swarm but not the time instant of their arrival. Thereby, a request was sent to the tracker every 400 milliseconds and it is considered that a peer has arrived for the first time at the time interval it got firstly listed.

6.2.2 Simulation Setup

According to Akamai's State of the Internet Q3 2016 report [59], the global average peak connection speed, which is considered to be more representative of the Internet connection capacity [1], is 37.2 Mb/s. Therefore, it was considered a star network topology with a central node mimicking an ISP, and with homogeneous leaf nodes connecting to it through asymmetric links: 30 Mbit/s downlink, 3 Mbit/s uplink, and 1 ms latency (2 ms delay between peers) to avoid any latency issues.

Peers communicate using TCP New Reno with an MTU of 1500 bytes, Maximum Segment Size (MSS) of 1460 bytes, and with Nagle's algorithm [41] disabled. Peers are provided with a list of all other peers currently in the swarm, request one block at a time, accept one request at a time, always attempt to complete the download in a single session, and leave immediately after completing the download. Version 3.23 of

Table 6.2: *Simulation setup for the Mistrustful P2P and traditional models.*

Configuration Variable	Mistrustful P2P	Traditional
Simulated Time	48 hours	
Network Topology	Star (central node mimicking an ISP)	
Peer Connections	30 Mbit/s downlink, 3 Mbit/s uplink	
End-to-end Latency	2 ms	
Transport Protocol	TCP New Reno	
MTU	1500 bytes	
MSS	1460 bytes	
Erasure Code	Storm	*n.a.*
Max. Swarm Backoff Time (μ_s)	$\bar{\tau}$	30 seconds
Max. Peer Backoff Time (μ_p)	$(k\bar{\tau}c)/m$	*n.a.*

ns-3 was used. Simulations were run for the first 48 hours of content sharing because the most demanding stage, *flash crowd*, usually ends within the first 36 hours. Thus, the simulation fully encompasses *flash crowd* stage and ends in *steady-state* stage.

Given that simultaneous chunk requests are not considered for the sake of clarity and tractability, it was defined a simplified model based on the BitTorrent protocol [@3] that achieves an average download bitrate that is, at least, in line with BitTorrent's. The peer roles and their sharing behavior were described in Section 1.2 (Traditional Peer-to-Peer Systems). The rarest first mechanism is simulated by letting the tracker keep a global list of the number of replicas of each chunk. Using this list, a peer picks randomly one of the ten rarest chunks (local rarity variation) and selects an available peer owning that chunk to send a request to. The set of peers cannot be periodically improved using the optimistic unchoking mechanism because there are no simultaneous downloads. As so, if the request fails because there are no peers available to provide that chunk, a backoff mechanism is enabled to prevent excessive protocol overhead. The backoff time increases linearly as a function of average chunk download time $\bar{\tau}$, never exceeding 30 seconds (typical optimistic unchoking period), and is given, in milliseconds, by $b = min\left(30000, \frac{\bar{\tau}}{4} \cdot u\right)$, where u is the number of consecutive failed requests. As for the Mistrustful P2P model, the actual backoff time is randomly generated within the interval $[0, b]$. Table 6.2 summarizes the simulation setup used for both models.

6.2.3 Experiments

84 experiments are considered to evaluate how content size, popularity (overall peer arrival rate), number of seeders, collusion size c, minimum network disguise overhead m, and cover downloads affect average download bitrate and download completion ratio; by experiment it is meant an evaluation using a distinct set of values for the variables in study.

Table 6.3: *Experiments considered for the evaluation of the Mistrustful P2P model and their categories.* The set of experiments for each category results from using different values for c, m, number of seeders, peer arrival traces – more popular (MP), popular (P), and less popular (LP) –, and content sizes. Over the first 48 hours, the total number of peer arrivals for MP, P, and LP traces are respectively 75800, 22700, and 3400 (approximately).

Category	Block Disclosure (m)	Collusion Size (c)	No. of Seeders	Peer Arrival Traces	Content Size (MiB)	No. of Experiments
Baseline	$k-1$	1, 31	1, 64	*MP*, *P* or *LP*	100, 800	24
Overhead	$k/2$	1, 31	1, 64	*MP*, *P* or *LP*	100, 800	24
Disguise	$k-1$ [a]	1, 31	1, 64	*MP*, *P* or *LP* [b]	100, 800	24
Traditional	n.a.	n.a.	1, 64	*MP*, *P* or *LP*	100, 800	12

[a] One third of the peers only downloads and shares half of the content to simulate a cover download.
[b] The mean peer arrival rates have a 50% increase to simulate arrivals due to cover downloads.

The experiments are divided into four categories: *baseline*, *overhead*, *disguise*, and *traditional*. The *baseline* category represents the less restrictive protection against any colluding group of a size up to c – all but one blocks can be disclosed to c peers ($m = k - 1$) –, and is used to estimate by comparison the impact of minimum network disguise overhead and cover content downloads on the sharing of genuine contents. The *overhead* category represents a reduced minimum network disguise overhead in which all peers fully download a genuine content but only disclose half of the blocks to any group of c peers ($m = k/2$). The *disguise* category represents the employment of a content interest disguise scheme where 50% more peers download 50% of the content; therefore, it is considered a 50% increase on the mean peer arrival rates (truncated to the nearest integer), and that one third of all peers only downloads 50% of content blocks before leaving. I.e., the number of active peers increases but also the resource usage. The *traditional* category represents traditional P2P systems, in which no privacy-enhancing mechanisms are employed, and is used as the performance reference.

For the first three categories 24 experiments are defined. For the last category the same experiments are defined except those for collusion size variants, totaling 12 experiments. It is considered the content size to be either 100 Mebibytes (MiB) or 800 MiB, the number of seeders to be either 1 or 64, and three video traces to compare different degrees of popularity: a more popular (MP), a popular (P), and a less popular (LP) contents. Over the first 48 hours, the total number of peer arrivals for MP, P, and LP traces are respectively 75800, 22700, and 3400 (approximately). Except for the last category, collusion size is either 1 or 31. Table 6.3 summarizes the experiments and their categories.

6.2.4 Results and Discussion

The main performance metric considered by P2P file sharing users is the average download time or the average download bitrate, which are two sides of the same

coin. The feasibility demonstration of block advertisement avoidance is conducted as follows. First, the Mistrustful P2P model is compared against the traditional P2P model for the case of minimum privacy protection (Figure 6.1), and the download completion ratio for each single experiment is detailed (Table 6.4). Then, the results obtained for all *baseline* experiments (Figure 6.2) are discussed to assess the impact of collusion size, content size, number of seeders, and content popularity, which are the variables that are expected to change more often. Lastly, the experiments of *baseline* category are compared with the experiments of *overhead*, *disguise* and *traditional* categories (Figures 6.3, 6.4 and 6.5) to, respectively, evaluate the impact of minimum network disguise overhead, estimate the impact of cover downloads (additional peer arrivals and partial downloads), and to assess the impact of the provided privacy protection as a whole. The results provided for each experiment are the average of four simulations, and the 95% confidence intervals are represented in all figures, although they are too small to be noticed in Figures 6.1, and 6.2.

Figure 6.1 provides a comparison of the average download bitrate achieved by the traditional P2P model and by the Mistrustful P2P model, which is set for minimum protection – *baseline* experiments for single peer attacks – given that traditional P2P systems do not provide any privacy protection. The results obtained by both models are equivalent, despite considering an optimistic model for representing traditional P2P systems, and show that peers are able to timely download contents without advertising what they download. The performance difference is negligible when considering single seeder experiments, and more noticeable for some experiments when considering 64 seeders. The benefit provided by the larger number of seeders seems to be dependent on the ratio between seeders and regular peers because it fades as the number of simultaneous peers increases, be it due to higher peer arrival rate or larger content size that requires peers to stay longer to complete their download. This correlation with the peer arrival rate is more evident for the 800 MiB content using the popular peer arrival trace (center right).

Table 6.4 provides the number and ratio of downloads completed for all 84 experiments, and shows that peers are able to complete their downloads: the download completion ratio is always above 96%. This ratio is below 100% for all experiments because there is always a set of peers that is unable to complete the download: peers arriving when the time left to end the simulation is less than the time required to complete the download. Thus, the 800 MiB experiments achieve a lower download completion ratio than their 100 MiB counterparts. The download completion ratio is identical for all categories, but the number of downloads is different for the *disguise* category because the peers used to emulate cover downloads (one third of all peers) are not considered as they only download partially the content (50%). The lowest download completion ratio is achieved for the popular (P) peer arrival trace, in particular for 800 MiB content size, due to an increase on the peer arrival rate near the end of the simulation (see Figure 6.1).

Table 6.4: *Number and ratio of downloads completed for all 84 experiments.* The experiments are presented grouped by peer arrival trace – more popular (MP), popular (P), and less popular (LP) –, content size (100 and 800 MiB), and category – *baseline*, *overhead*, *disguise* and *traditional* –, for either 1 or 64 seeders, and a collusion of either 1 or 31.

Peer Arrival Trace	Content Size (MiB)	Category	1 Seeder, Col. of 1	1 Seeder, Col. of 31	64 Seeders, Col. of 1	64 Seeders, Col. of 31
MP	100	*Baseline*	75313 (99.84%)	75314 (99.84%)	75325 (99.86%)	75325 (99.86%)
		Overhead	75314 (99.84%)	75313 (99.84%)	75325 (99.86%)	75324 (99.86%)
		Disguise	75583 (99.89%)	75581 (99.88%)	75595 (99.90%)	75596 (99.90%)
		Traditional	75317 (99.85%)	–	75343 (99.88%)	–
	800	*Baseline*	74600 (98.90%)	74596 (98.89%)	74622 (98.93%)	74622 (98.93%)
		Overhead	74595 (98.89%)	74599 (98.90%)	74619 (98.92%)	74623 (98.93%)
		Disguise	74888 (98.97%)	74883 (98.96%)	74909 (98.99%)	74910 (99.00%)
		Traditional	74618 (98.92%)	–	74636 (98.94%)	–
P	100	*Baseline*	22462 (99.52%)	22461 (99.52%)	22473 (99.57%)	22472 (99.57%)
		Overhead	22462 (99.52%)	22461 (99.52%)	22473 (99.58%)	22473 (99.57%)
		Disguise	22516 (99.58%)	22515 (99.58%)	22524 (99.62%)	22524 (99.62%)
		Traditional	22467 (99.55%)	–	22491 (99.65%)	–
	800	*Baseline*	21831 (96.73%)	21830 (96.73%)	21858 (96.85%)	21855 (96.83%)
		Overhead	21831 (96.73%)	21833 (96.74%)	21858 (96.85%)	21855 (96.83%)
		Disguise	21856 (96.67%)	21858 (96.67%)	21877 (96.76%)	21877 (96.76%)
		Traditional	21843 (96.78%)	–	21871 (96.91%)	–
LP	100	*Baseline*	3308 (99.91%)	3263 (98.54%)	3308 (99.91%)	3308 (99.91%)
		Overhead	3308 (99.91%)	3237 (97.77%)	3308 (99.91%)	3308 (99.91%)
		Disguise	3298 (99.97%)	3259 (98.80%)	3298 (99.97%)	3298 (99.97%)
		Traditional	3308 (99.91%)	–	3308 (99.91%)	–
	800	*Baseline*	3259 (98.41%)	3250 (98.14%)	3269 (98.72%)	3269 (98.72%)
		Overhead	3260 (98.44%)	3216 (97.14%)	3269 (98.72%)	3269 (98.72%)
		Disguise	3244 (98.34%)	3238 (98.15%)	3252 (98.58%)	3252 (98.57%)
		Traditional	3260 (98.46%)	–	3276 (98.94%)	–

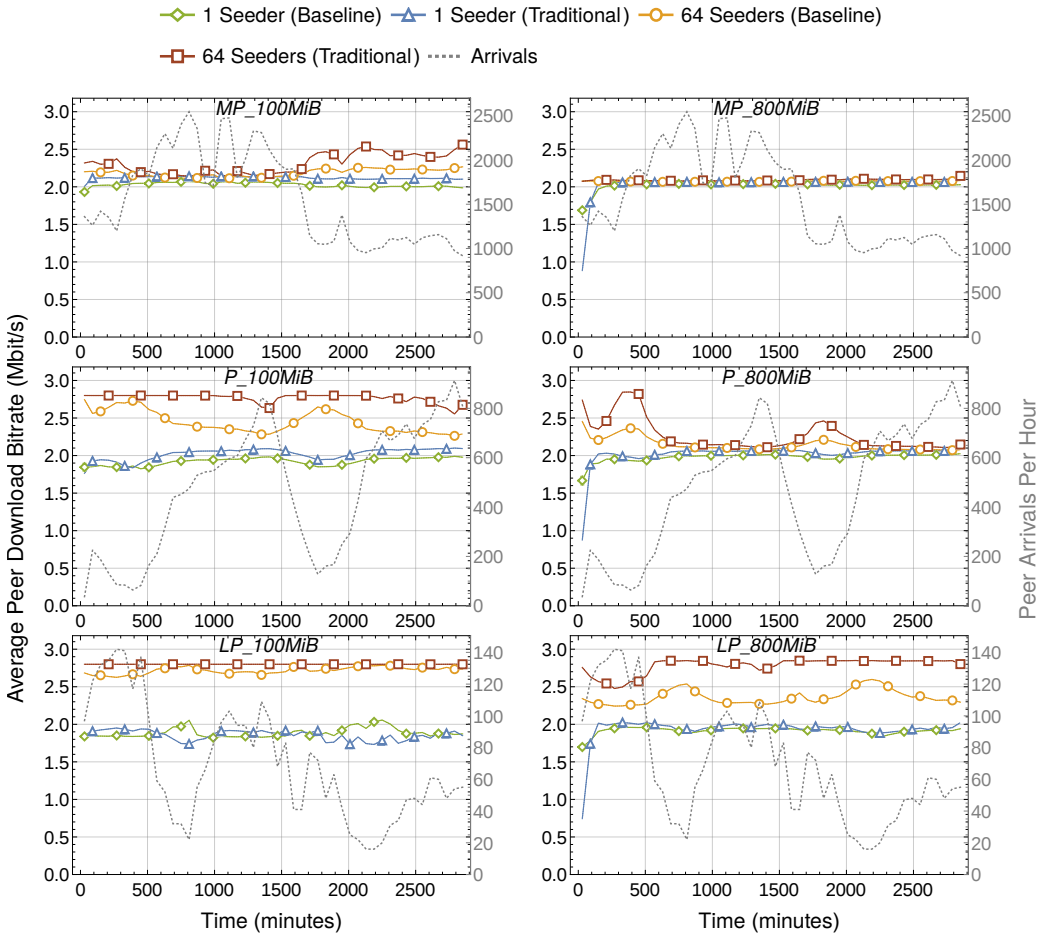


Figure 6.1: *Average download bitrate over one hour periods for 100 MiB (left) and 800 MiB (right) contents using a more popular (MP), a popular (P), and a less popular (LP) peer arrival traces as input (one per row).* Each plot depicts four experiments that are a result of using either 1 or 64 seeders, and considering either the Mistrustful P2P model (*baseline* experiments of single peer attacks) or the traditional P2P model. The peer arrival rate is represented by a dotted gray line with a y-scale on the right.

All *baseline* experiments are depicted in Figure 6.2 to assess the impact of collusion size, content size, number of seeders, and content popularity. A collusion size of up to 31 peers requires a peer to contact, at least, 32 unique peers to be able to complete the download. Thereby, when considering 64 seeders, the results are identical for both a collusion of 1 and of 31 because it is guaranteed that there are always enough peers available. However, in the experiments considering a single seeder and a collusion of 31, mainly for smaller and less popular contents as there are less simultaneous peers, the correlation between the average download bitrate and the peer arrival rate is evident. As the number of simultaneous peers increases, be it due to higher peer arrival rate or larger content size, the performance gap between collusion of 1 and collusion of 31 for single seeder experiments fades until it becomes negligible. Thus, single seeder and collusion of 31 experiments are used to compare the performance of *baseline* category with all others, given that they provide the worst case and enable a better assessment of the impact of each variable.

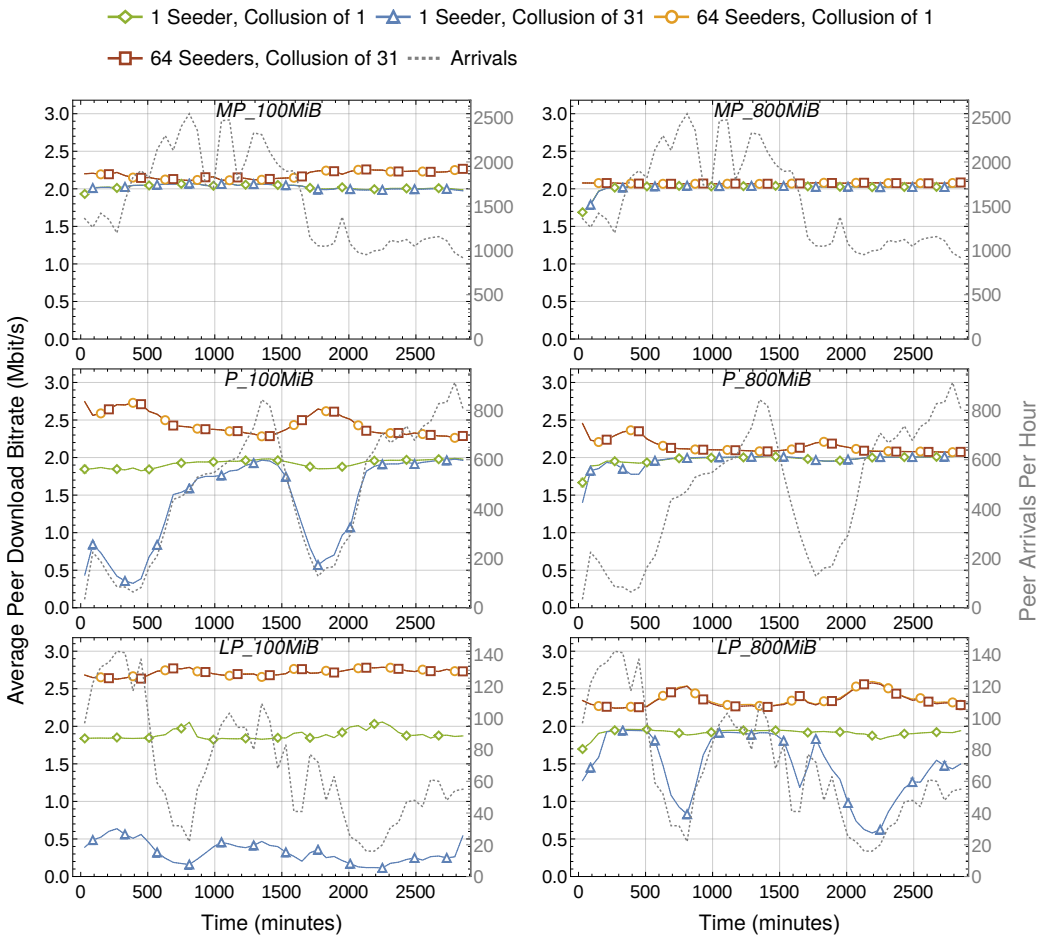


Figure 6.2: *Average download bitrate over one hour periods for all baseline experiments.* Contents have either 100 MiB (left) or 800 MiB (right) and use a more popular (*MP*), a popular (*P*), and a less popular (*LP*) peer arrival traces as input (one per row). Each plot depicts four experiments that are a result of using either 1 or 64 seeders, and considering either single peer attacks or collusion attacks of, at most, 31 peers. The peer arrival rate is represented by a dotted gray line with a y-scale on the right.

Figures 6.3, 6.4, and 6.5 provide the results obtained in each category – *baseline*, *overhead*, *disguise*, and *traditional* – for, respectively, the ratio of block requests sent to seeders out of all block requests, the average download bitrate, and the average ratio of time spent on backoff (average backoff time ratio). As shown by Figure 6.3, the main reason for the performance gap between the Mistrustful P2P model, for all experiments using the less popular peer arrival trace, and the traditional P2P model is the exploitation of the seeder. Unlike the traditional P2P model that favors requests to seeders, the Mistrustful P2P model treats all peers alike and, on average, shares no more than $^{m}/_{c}$ blocks with each individual peer, including the seeder. For this reason, it presents a low ratio of block requests sent to seeders that is not subject to significant variations. The increase of the confidence intervals in Figures 6.4, and 6.5 for the 800 MiB contents, in particular at the beginning of the most popular content and at the end of the less popular one, is mainly due to the heterogeneous perception of block availability (request successfulness) between peers. Contents that take more time to download, be it due to their larger size or

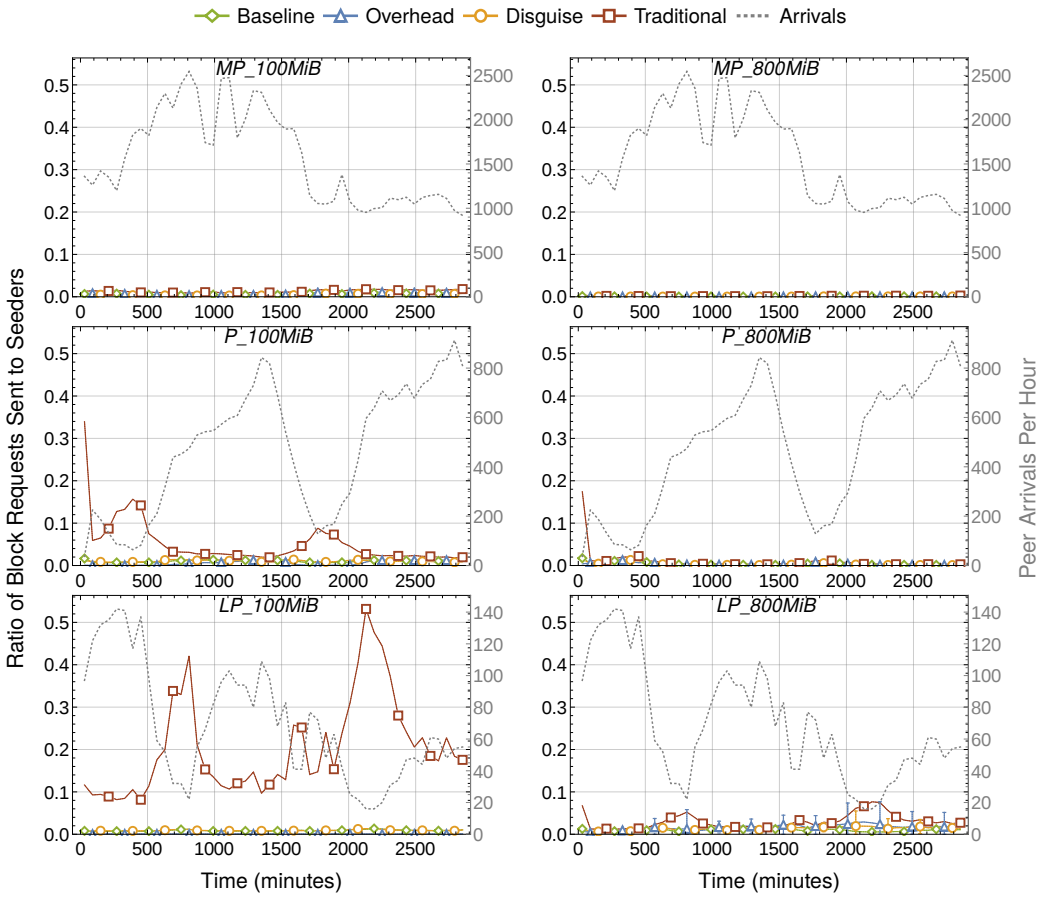


Figure 6.3: *Average ratio of requests sent to seeders over one hour periods for 100 MiB (left) and 800 MiB (right) contents using a more popular (MP), a popular (P), and a less popular (LP) peer arrival traces as input (one per row).* Each plot depicts each experiment category – *baseline*, *overhead*, *disguise*, and *traditional* – for a single seeder and collusion attacks of, at most, 31 peers (except *traditional*). The peer arrival rate is represented by a dotted gray line with a y-scale on the right.

due to the lower availability of useful blocks, are more likely to create heterogeneity in the number of downloaded blocks between peers. Thus, the perception of content availability diverges significantly between the peers beginning and those concluding the download, which is reflected on the observed backoff time.

In the *overhead* experiments, peers can disclose, at most, 50% of the blocks they download to any set of c peers ($m = {}^{k}/_{2}$) hardening the probability of retrieving useful blocks, in particular during the *flash crowd* stage because the few useful blocks are being uploaded mostly by seeders. Given that all contents are divided into 64 blocks, increasing the content size augments this effect as seeders will take more time to share the blocks required. This effect is noticeable for the most popular content in Figure 6.4 but not for the less popular content because the main bottleneck with the former is the unavailability of blocks and not the peer unavailability as with the latter. As shown by Figure 6.5, apart from the beginning, the average backoff time ratio is low throughout the content sharing of the more popular content while it is high throughout the content sharing of the less popular one. Therefore, it can be

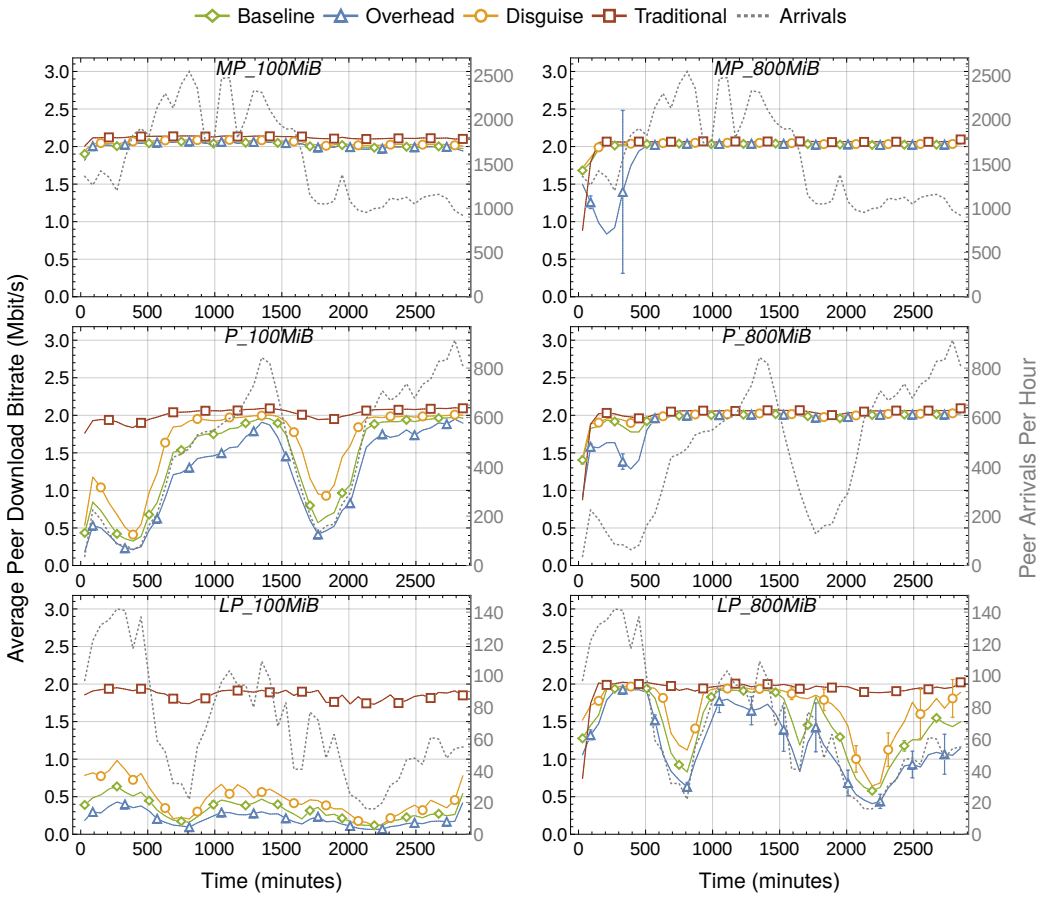


Figure 6.4: *Average download bitrate over one hour periods for 100 MiB (left) and 800 MiB (right) contents using a more popular (MP), a popular (P), and a less popular (LP) peer arrival traces as input (one per row).* Each plot depicts each experiment category – *baseline*, *overhead*, *disguise*, and *traditional* – for a single seeder and collusion attacks of, at most, 31 peers (except *traditional*). The peer arrival rate is represented by a dotted gray line with a y-scale on the right.

concluded that the impact of minimum network disguise overhead, as for collusion size, depends on the number of seeders and simultaneous peers. Its impact dilutes as the number of simultaneous peers increases.

In the *disguise* experiments, it was considered a 50% increase of the peer arrival rate due to cover downloads and that those additional peers only download and share 50% of the blocks required to complete the download. As for all other peers, they leave immediately once they download the content. The results obtained indicate that cover downloads improve the performance of the Mistrustful P2P model on all experiments. Still, this is just an estimation that highly depends on the scheme being used to select cover contents and how much of each to download in order to disguise user content interests.

In the *traditional* experiments, despite considering an optimistic model that has global knowledge of block availability/rarity and that is able to take more advantage of seeders, around 20% of time is still spent on backoff when used with less popular contents. This highlights the importance of the number of simultaneous

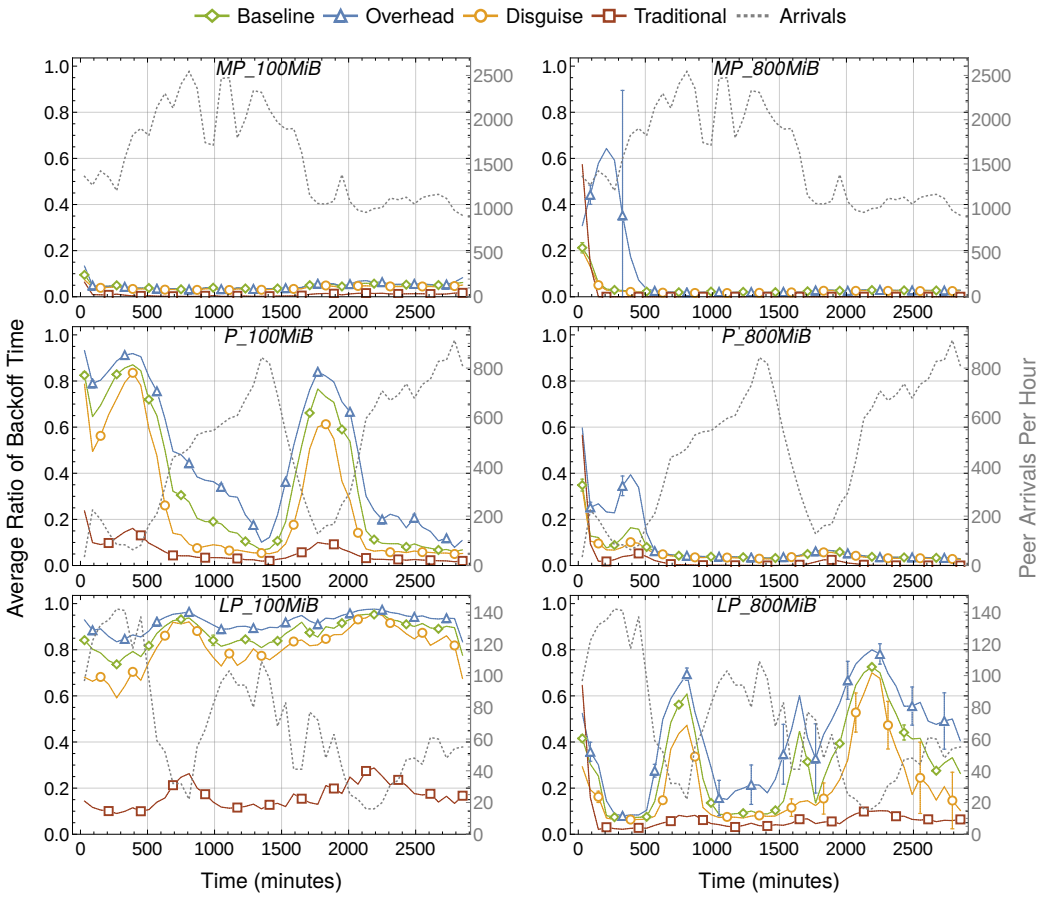


Figure 6.5: *Average ratio of backoff time over one hour periods for 100 MiB (left) and 800 MiB (right) contents using a more popular (MP), a popular (P), and a less popular (LP) peer arrival traces as input (one per row).* Each plot depicts each experiment category – *baseline*, *overhead*, *disguise*, and *traditional* – for a single seeder and collusion attacks of, at most, 31 peers (except *traditional*). The peer arrival rate is represented by a dotted gray line with a y-scale on the right.

peers on the overall performance, which is amplified by the Mistrustful P2P model when considering protection against larger attackers, given that peers need to wait for other peers to join before being able to complete the download (temporary unavailability of peers).

6.3 Conclusions

The Mistrustful P2P model provides deterministic protection of user content interests against attacks of a size up to a configured level, has no trust requirements, and prevents user legal liability in case of legitimate usage. The privacy claims of the Mistrustful P2P model were validated through a security analysis, which, due to the multitude and complexity of copyright laws across the globe, considers that a legitimate user should not be held legally liable as long as the main motivation to use a P2P system is to share legit contents, and it cannot be proven that the user had access to the content data.

The performance evaluation shows that the Mistrustful P2P model is feasible: peers are able to timely download contents without advertising what they download. Its performance, when considering minimum protection ($c = 1$, and $m = k - 1$), is close to the one of the optimistic model that represents traditional P2P file sharing systems. The impact of stronger protection depends significantly on the number of simultaneous peers and, after the *flash crowd* stage, becomes negligible for popular contents. For contents with similar size and popularity, the number of simultaneous peers can be increased either by adding more seeders, which also help to improve the distribution and diversity of blocks across the network, or by having peers to download cover contents. Although dependent on the scheme used for content interest disguise, cover downloads are expected to improve the overall performance.

The feasibility of the Mistrustful P2P model was demonstrated through simulation considering the most demanding stage of the content download process: the *flash crowd*. The peer arrivals are based on real traces, and their number ranges from few thousands (less popular content trace) up to several tens of thousands (most popular content trace). Even using CIDRarchy and applying several optimizations, simulating a single experiment can take up to three days on an Intel Xeon E5-2643v4@3.40GHz CPU.

7 Conclusion

> *If, after I die, people want to write my biography, there is nothing simpler. They only need two dates: the date of my birth and the date of my death. Between one and another, every day is mine.*
>
> **— Fernando Pessoa**

The continuous advances in computer technology over the last decades have enabled organizations and individuals to collect information about users, combine facts from different sources, and merge them to create databases of personal information that were previously impossible to set up [66]. Users cannot simply rely on the difficulty of gathering information to preserve their privacy, they need to preserve it through active means. P2P networks are an important source of personal information, especially when traditional P2P file sharing systems such as BitTorrent are used, because P2P networks are extensively used for large-scale file sharing, user content interests may be trivially identified by simply gathering the list of peers registered at the tracker(s), and the download of a content, in part or fully, can be trivially proved given that peers are required to advertise what they download. Legitimate users may also be subject to legal liability as a result of, unknowingly and unwillingly, downloading an illegal content that has either been published with a misleading description or whose hash has been obtained through an insidious resource. Moreover, it is estimated that, over the course of a month, 96.3% of users of BitTorrent portals have downloaded at least one infringing content [47], weakening the plausible deniability of legitimate users in the face of evidences.

Privacy-preserving P2P file sharing systems that address the content interest identification issues through content interest disguise – by downloading additional contents of no interest to the user in order to hide his real interests – still require peers to advertise, either fully or partially, what they download. Furthermore, any legal issues that may arise from their legit and well-intended use are not addressed. Users have adopted anonymity systems such as Tor due to the lack of alternatives [8], misunderstanding the privacy guarantees provided by such systems [9], in particular when relaying traffic of insecure applications [35] such as BitTorrent. Anonymity systems provide a channel to anonymously transmit messages, the content of such messages and any information that they may disclose is the sole responsibility of the application being used. This thesis proposed and evaluated a novel P2P file sharing model, the Mistrustful P2P model, that addresses these issues.

Chapter 1 provided the legal and ethical framework for P2P file sharing, and detailed the problem this work aimed to solve, its main contributions and the working hypothesis: *It is possible to deterministically hide the content interests of users sharing publicly accessible files over untrusted P2P networks without disclosing what peers download or miss.* Chapter 2 described the related work for each one of the three main contributions, which were presented in the following three chapters, respectively, the Mistrustful P2P model (Chapter 3), Storm erasure codes (Chapter 4), and CIDRarchy ns-3 routing protocol (Chapter 5). The validation of the Mistrustful P2P model, the core contribution, was conducted in Chapter 6; Storm and CIDRachy were validated in the chapters they were presented.

The remainder of this chapter is structured as follows. Section 7.1 presents the ethical and legal considerations. Sections 7.2, 7.3, and 7.4 draw the overall conclusions for, respectively, user content interest concealment, the employment of erasure codes, and large-scale simulation of Internet systems. Section 7.5 acknowledges some known limitations of the Mistrustful P2P model, and Section 7.6 presents the future work. The concluding remarks are provided in Section 7.7.

7.1 Ethical and Legal Considerations

There are no universal ethical guidelines that can be followed because ethics reflect the unique existential experiences that are accumulated as individuals as well as societies, which necessarily differ between individuals, nations, religions, and cultures as a result of differing experiences and interpretations [6]. P2P networks connect users without considering international borders, religion, culture or race, but there is the risk of having contents available that are considered ethical by a user and unethical by another user. In such case, either the set of available contents is limited to those that are universally considered ethical, which may render the set empty, or each content needs to be classified individually to enable filtering according to one's ethics.

The multitude and complexity of laws across the globe also reflect these differences, and thus make it impossible to define clear legal liability boundaries. Asian traditions of collective ownership conflict with Western protection of intellectual property, and many of the ways the former use software is considered software piracy by the latter [66]. Moreover, the copyright rights are not granted worldwide (see Berne Convention [@1]), the private copying law, when present, introduces an exception to these exclusive rights under some conditions, it is hard to determine the applicable legal framework when the content sharing crosses multiple jurisdictions, and, even when the applicable legal framework can be determined, the copyright rights are subject to proportionality assessment if they conflict with other interests and rights in order to balance all the rights and interests at stake [30].

Although subject to different interpretations, the basic ethical imperatives are that a person should not, knowingly and willingly, cooperate in or contribute to the wrongdoing of another, and that the human intellectual creativity needs to be encouraged and stimulated in order to promote progress. The extent of the incentives depends on the cultural background as the well-being of the society may be incentive enough (collective ownership) or further incentives may be required (intellectual property rights). As so, the Mistrustful P2P model was designed considering that the users of P2P systems should not be held legally liable for, unknowingly and unwillingly, downloading an illegal content (direct liability) or for contributing to its download (indirect liability) as long as: the main motivation to use the P2P system is to share legit contents; it cannot be proven that the user had access, either in part or fully, to the content data. Under such conditions, it is plausible and likely that the user has been mislead into downloading an illegal content because the system is mostly used to share legit contents, and there is no direct benefit in downloading a content whose data cannot be accessed other than hiding user content interests.

7.2 Hiding User Content Interests

The reasons behind seeking privacy can be legit, such as to avoid user profiling or to privately download contents that are considered embarrassing or objectionable from a given point-of-view, but may also be illicit, such as to download illegal contents. In order to better protect legitimate and well-intended users, the Mistrustful P2P model attempts to find the proper balance between privacy, accountability, and performance.

The Mistrustful P2P model hides user content interests through content interest disguise. It provides plausible deniability to the user, has no trust requirements, thus enables content sharing over untrusted P2P networks, and ensures deterministic protection of user content interests against passive attacks of any size and against active attacks of a size up to a configured level c (see Section 6.1.2). Peers do not advertise what they download, defeating passive attacks, and forcing attackers to engage in the content sharing thus increasing the resources required to launch attacks. Genuine and cover contents cannot be distinguished, up to a configured level c, because at least m blocks are downloaded per cover content, and at most, m blocks are implicitly disclosed to any set of c peers.

The main motivation to use a system based on the Mistrustful P2P model must be to share legit contents, otherwise the first condition to prevent user legal liability is not fulfilled. It is of utmost importance to discourage illegal content download and make illegitimate users accountable, i.e., publishers, seeders and commoners; trackers only provide the set of peers in a given swarm, and thus have no active role in this matter. The second condition is fulfilled if there is no proof of content access. Such system is able to make users accountable because publishers and seeders forgo their privacy, peers communicate directly, and the set of downloaded contents (both

genuine and cover contents) is not concealed. Publishers and seeders are interested in disseminating a content, and thereby they do not require the concealment of their content interests, which enables to ban those misbehaving. Commoners cannot hide behind other peers as they communicate directly, and the strength of their plausible deniability directly depends on the number of illegal but non-misleading contents downloaded: it weakens as their number increases. The proof of content access is avoided by only granting access to content data after full content download – chunks are encoded in a way that enables decoding only after full download –, and by avoiding proof of full content download – constraining the set of blocks implicitly disclosed to other peers (see Section 6.1.3).

There is a performance trade-off between privacy and performance, and therefore the user is able to configure the size c of the largest group of colluding peers to be protected against, and the minimum network overhead m of content interest disguise. On average, m/c blocks can be disclosed to each peer, and thus the average download bitrate increases as this ratio increases (the amount of available requests increases). The extent of the variation depends on the content popularity because, despite the amount of available block requests per peer being the same, the total amount of available block requests also increases as more peers are available. The performance evaluation (Section 6.2) has shown that the Mistrustful P2P model is feasible, and, when considering minimum protection ($c = 1$, and $m = k - 1$), its performance is close to the one of traditional P2P file sharing systems.

7.3 Erasure Codes

The erasure coding mechanism of the Mistrustful P2P model uses erasure codes to enable the probability of randomly retrieving a block to become significant, and to enable decoding only after full download. Any rateless erasure code may be used, so that dynamic generation of new blocks as needed is enabled, and, if it is systematic or enables decoding before full download, the content needs to be encrypted before sharing. The network is typically the most constrained P2P file sharing resource, not the CPU [36], so MDS erasure codes are more suitable for P2P file sharing because they introduce no network overhead. For less common scenarios, non-MDS erasure codes may be more suitable because they reduce the encoding and decoding time complexities by introducing network overhead. In either case, if the erasure codes are non-systematic and enable decoding only after full download, additional computational overhead due to encryption is avoided.

To the best of the author's knowledge, the MDS erasure codes available in the literature were either fixed-rate or had high encoding and decoding time complexities (see Section 2.2), and their practical use was also limited to about 2^{16}. Therefore, Storm erasure codes were proposed to fill this gap: they are rateless MDS erasure codes based on RS codes with $\Theta(n \log k)$ encoding time complexity and $\min\left\{n \log n, k \log^2 k\right\}$ upper bound for decoding time complexity. Their perfor-

mance was assessed over $\mathbb{F}_{(2^{31}-1)^2}$ and compared against Soro et al.'s [57] – the only ones with $\Theta(n \log n)$ time complexity that admit any power of two for n and k, $k \leq n \leq 2^{16}$, achieving nearly twice the throughput of equivalent codes. Unlike Soro et al.'s that are defined over Fermat finite fields and are thereby limited to 2^{16}, Storm erasure codes are defined over complex Mersenne finite fields, which have no known size limit. Therefore, if $\mathbb{F}_{(2^{31}-1)^2}$ is considered insufficient ($k \leq n \leq 2^{32}$), a larger one such as $\mathbb{F}_{(2^{127}-1)^2}$ can be used ($k \leq n \leq 2^{128}$).

7.4 Large-scale Simulation of Internet Systems

For large-scale Internet systems such as P2P networks, which usually involve thousands or even millions of peers, it becomes impracticable to test accurately the designed protocols either analytically or using real large-scale implementations, and small scale tests may not be enough as some issues may only arise at the scale of thousands of peers or more [10]. Therefore, simulation plays a vital role on designing, building, understanding and thoroughly evaluating large-scale Internet systems, but simulators are usually forced to trade off simulation accuracy for scale [20] because it is hard to evaluate Internet systems over a large and complex Internet topology while using a complex and realistic network stack, and its protocols.

ns-3 [@4] is the successor of ns-2, the most popular network simulator for research [34], and simulates realistically the network stack. Still, the scale and complexity of an Internet-like topology is limited by the IP forwarding of existing ns-3 routing protocol implementations (see Section 2.3), given that they are generic and thus cannot take advantage of the hierarchical structure of Internet-like networks. CIDRarchy ns-3 routing protocol was proposed to take advantage of this hierarchical structure, and thus enable large-scale Internet-like network simulation by performing IP packet forwarding in constant time. It provides automatic IPv4 address assignment, and a helper to enable complex topology creation with a few lines of code and also the addition of new hosts at runtime. CIDRarchy was implemented and evaluated in ns-3 simulator, and the simulation time gains over existing ns-3 routing protocols can reach over one order of magnitude (see Section 5.4). CIDRarchy greatly reduces the time spent in IPv4 forwarding, and a simulation taking one month to complete may require less than four days to complete if using it.

Despite the merits of ns-3, at the time of writing, the current development version of the ns-3 simulator (3.27) is still trying to close several open issues of its TCP models [@5]. Some of these issues were experienced in first hand during the validation of the Mistrustful P2P model, and, although they do not affect the quality of the results, patching and additional runtime checks were required to prevent high memory usage and to ensure proper simulation ending. These issues may not manifest for small-scale tests, but will probably manifest in simulations in which there is a high number of TCP connections being created and closed, such as the ones ran to validate the Mistrustful P2P model.

7.5 Known Limitations

The Mistrustful P2P is a model, not a full-featured privacy-preserving P2P system, and thereby some building blocks have been instantiated just to prove the working hypothesis while other building blocks such as the disguise scheme are yet to be defined. The main known limitations already identified are the following:

- External entities monitoring all traffic of a peer, such as ISPs or governments, are able to identify user content interests. Protection against link monitoring can be achieved by encrypting communications between peers, but it requires key exchange and distribution mechanisms, which are out of the scope of this work;
- The size c of the largest colluding group that can be considered is limited by both the minimum network overhead m of content interest disguise and by the number k of chunks into which the content is partitioned: $c \leq m < k$. Although $c < k$ is required to prevent proof of download, the condition $c \leq m$ may eventually be relaxed, e.g., by constraining also the number of blocks that are disclosed to any set of c peers among all contents (both genuine and cover) so that, at most, an attacker is able to identify a small percentage of cover contents;
- The seeders are required to keep sharing new blocks during the lifetime of a content, otherwise the distribution of the content may harden or even preclude. The current instantiation of the block selection mechanism also contributes to this because the weight attributed to a block decreases monotonically, and therefore older blocks tend to become unavailable. As it is currently, and depending on whether the publisher intends to control the lifetime of the content or not, it can be considered either as a feature or as a limitation: publishers can control for how long the content is distributed but more resources are required to keep the content available.

7.6 Future Work

There is still a long path to cross in order to create a privacy-preserving P2P system on top of the Mistrustful P2P model, but the next steps are the following:

- **Define the disguise scheme.** The disguise scheme needs to be defined as well as all underlying mechanisms that it requires (e.g., content classification). It is responsible for selecting which cover contents should be used according to user preferences, how much to download of each, and must also thwart attacks based on multiple session analysis. Less popular contents should be favored to improve their performance, unless privacy issues are raised.

- **Conduct a performance evaluation for larger values of k.** For the validation of the Mistrustful P2P model, due to the large amount of variables being evaluated, contents were considered to be partitioned into 64 blocks ($k = 64$). It is important to evaluate the impact of using greater values for k so that larger colluding groups can be considered.
- **Parallelize Storm erasure codes.** The performance evaluation of Storm erasure codes was conducted on an Intel Core i5-560M using a single-threaded implementation. Thus, a parallel multi-core implementation will be created as there is room for significant performance improvements. Also, their performance will be evaluated on mobile terminals.
- **Improve the instantiation of several mechanisms.** The block selection mechanism assesses the rarity/popularity of a block using only the feedback information provided by incoming block requests, but it will also consider the feedback information provided by outgoing block requests. For the sake of clarity and simplicity, the peer selection mechanism was defined to be random. Selecting peers from different countries or using some heuristic to select peers that are less likely to be related will be considered, as it may increase the probability of protecting user's privacy against underestimated attackers.

7.7 Concluding Remarks

The working hypothesis has been proven, and the proposed model, Mistrustful P2P, addresses the four points of the problem definition: 1) hide user content interests; 2) have no trust requirements; 3) prevent user liability; 4) enable timely downloads. The Mistrustful P2P model deterministically hides user content interests against attacks of a size up to a configured level. It has no trust requirements, prevents legal liability, and its performance evaluation shows that peers are able to timely download contents without advertising what they download. Moreover, when considering minimum protection, its performance is close to the one of traditional P2P file sharing systems.

This thesis has three main contributions – Mistrustful P2P model, Storm erasure codes, and CIDRarchy ns-3 routing protocol –, and four papers were produced as a direct result of it:

[1] Silva, P. M. da, Dias, J., and Ricardo, M. "CIDRarchy: CIDR-based ns-3 Routing Protocol for Large Scale Network Simulation". In: *Proceedings of the 8^{th} International Conference on Simulation Tools and Techniques*. SIMUTools '15. Athens, Greece, 2015, pp. 267–272.

[2] Silva, P. M. da, Dias, J., and Ricardo, M. "Storm: Rateless MDS Erasure Codes". In: *Wireless Internet: 8^{th} International Conference, WICON 2014, Lisbon, Portugal, November 13-14, 2014, Revised Selected Papers*. Ed. by

Mumtaz, S., Rodriguez, J., Katz, M., Wang, C., and Nascimento, A. Springer International Publishing, 2015, pp. 153–158.

[3] Silva, P. M. da, Dias, J., and Ricardo, M. “Mistrustful P2P: Privacy-preserving File Sharing Over Untrustworthy Peer-to-Peer Networks”. In: *Proceedings of IFIP Networking 2016*. IFIP Networking ’16. Vienna, Austria, 2016, pp. 395–403.

[4] Silva, P. M. da, Dias, J., and Ricardo, M. “Mistrustful P2P: Deterministic Privacy-preserving P2P File Sharing Model to Hide User Content Interests in Untrusted Peer-to-Peer Networks”. In: *Computer Networks* 120 (2017), pp. 87–104.

BIBLIOGRAPHY

[1] Akamai. *State of the Internet Metrics: What Do They Mean?* (Cit. on p. 68).

[2] Bauer, K., McCoy, D., Grunwald, D., and Sicker, D. "BitBlender: Light-Weight Anonymity for BitTorrent". In: *Proceedings of the Workshop on Applications of Private and Anonymous Communications (AlPACa 2008)*. Istanbul, Turkey: ACM, Sept. 2008 (cit. on pp. 2, 12, 13).

[3] Betker, A., Gamrath, I., Kosiankowski, D., et al. "Comprehensive topology and traffic model of a nationwide telecommunication network". In: *Optical Communications and Networking, IEEE/OSA Journal of* 6.11 (Nov. 2014), pp. 1038–1047 (cit. on pp. 49, 50, 52).

[4] Borders, R. W. *Enemies of The Internet 2014*. Tech. rep. (cit. on p. 3).

[5] Borodin, A. and Moenck, R. "Fast modular transforms". In: *J. Comput. Syst. Sci.* 8.3 (June 1974), pp. 366–386 (cit. on pp. 16, 41).

[6] Capurro, R. "Intercultural Information Ethics". In: *The Handbook of Information and Computer Ethics*. John Wiley & Sons, Inc., 2009, pp. 639–665 (cit. on pp. 3, 80).

[7] Carbunaru, C., Teo, Y. M., Leong, B., and Ho, T. "Modeling Flash Crowd Performance in Peer-to-Peer File Distribution". In: *IEEE Transactions on Parallel and Distributed Systems* 25.10 (Oct. 2014), pp. 2617–2626 (cit. on p. 68).

[8] Chaabane, A., Manils, P., and Kaafar, M. "Digging into Anonymous Traffic: A Deep Analysis of the Tor Anonymizing Network". In: *Network and System Security (NSS), 4^{th} International Conference on*. 2010, pp. 167–174 (cit. on pp. 2, 79).

[9] Chakravarty, S., Portokalidis, G., Polychronakis, M., and Keromytis, A. "Detection and analysis of eavesdropping in anonymous communication networks". In: *International Journal of Information Security* 14.3 (2015), pp. 205–220 (cit. on pp. 2, 79).

[10] Cheng, L., Hutchinson, N., and Ito, M. "P2PNet: A Simulation Architecture for Large-scale P2P systems". English. In: *New Technologies, Mobility and Security*. Ed. by Labiod, H. and Badra, M. Springer Netherlands, 2007, pp. 567–581 (cit. on pp. 8, 49, 83).

[11] Choffnes, D. R., Duch, J., Malmgren, D., et al. *SwarmScreen: Privacy Through Plausible Deniability in P2P Systems*. Tech. rep. Northwestern EECS, Mar. 2009 (cit. on pp. 2, 12, 13, 23).

[12] Chu, E. and George, A. *Inside the FFT Black Box: Serial and Parallel Fast Fourier Transform Algorithms*. Computational Mathematics. Taylor & Francis, 1999 (cit. on p. 42).

[13] Clough, J. *Principles of Cybercrime*. 1^{st}. New York, NY, USA: Cambridge University Press, 2010 (cit. on p. 3).

[14] Cohen, B. “Incentives build robustness in BitTorrent”. In: *Workshop on Economics of Peer-to-Peer systems*. Vol. 6. 2003, pp. 68–72 (cit. on p. 6).

[15] Crandall, R. E. and Pomerance, C. *Prime Numbers: A Computational Perspective. Second Edition*. Springer, 2005, pp. 509–518 (cit. on pp. 43, 44).

[16] Creutzburg, R. and Tasche, M. “Parameter Determination for Complex Number-Theoretic Transforms Using Cyclotomic Polynomials”. In: *Mathematics of Computation* 52.185 (1989), pp. 189–200 (cit. on p. 41).

[17] Didier, F. “Efficient erasure decoding of Reed-Solomon codes”. In: *CoRR* abs/0901.1886 (2009) (cit. on p. 17).

[18] Dingledine, R., Mathewson, N., and Syverson, P. “Tor: The Second-generation Onion Router”. In: *Proceedings of the* 13^{th} *Conference on USENIX Security Symposium - Volume 13*. SSYM’04. 2004 (cit. on pp. 1, 12).

[19] Douceur, J. R. “The Sybil Attack”. In: *Peer-to-Peer Systems: First International Workshop, IPTPS 2002*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 251–260 (cit. on p. 29).

[20] Eger, K., Hoßfeld, T., Binzenhöfer, A., and Kunzmann, G. “Efficient Simulation of Large-scale P2P Networks: Packet-level vs. Flow-level Simulations”. In: *Proceedings of the Second Workshop on Use of P2P, GRID and Agents for the Development of Content Networks*. UPGRADE ’07. Monterey, California, USA: ACM, 2007, pp. 9–16 (cit. on pp. 49, 83).

[21] Fuller, V. and Li, T. *Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan*. RFC4632. Aug. 2006 (cit. on p. 50).

[22] Gao, S. “A New Algorithm for Decoding Reed-Solomon Codes”. In: *Communications, Information and Network Security, V. Bhargava, H. V. Poor, V. Tarokh, and S. Yoon*. Kluwer, 2002, pp. 55–68 (cit. on pp. 16, 41).

[23] Goldschlag, D., Reed, M., and Syverson, P. “Onion Routing”. In: *Commun. ACM* 42.2 (Feb. 1999), pp. 39–41 (cit. on p. 12).

[24] Guo, L., Chen, S., Xiao, Z., et al. “A Performance Study of BitTorrent-like Peer-to-peer Systems”. In: *IEEE J.Sel. A. Commun.* 25.1 (Jan. 2007), pp. 155–169 (cit. on p. 67).

[25] Harrigan, K. and Riley, G. “Simulation Speedup of ns-3 Using Checkpoint and Restore”. In: *Proceedings of the 2014 Workshop on ns-3*. WNS3 ’14. Atlanta, Georgia: ACM, 2014, 7:1–7:7 (cit. on p. 50).

[26] Hazucha, B. *Private Copying and Harm to Authors: Compensation versus Remuneration*. `http://ssrn.com/abstract=2699070`. Dec. 2015 (cit. on pp. 3, 4).

[27] He, N., Xu, Y., Cao, J., et al. “ROME: Rateless Online MDS Code for Wireless Data Broadcasting”. In: *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*. Dec. 2010, pp. 1–5 (cit. on p. 16).

[28] He, Y., Siganos, G., and Faloutsos, M. “Internet Topology”. English. In: *Encyclopedia of Complexity and Systems Science*. Ed. by Meyers, R. A. Springer New York, 2009, pp. 4930–4947 (cit. on p. 49).

[29] Himma, K. E. and Tavani, H. T., eds. *The Handbook of Information and Computer Ethics*. John Wiley & Sons, Inc., 2009 (cit. on p. 3).

[30] Husberg, M. *Blocking injunction requisites - The balancing of rights and other aspects of blocking injunctions towards intermediaries*. Graduate thesis. 2015 (cit. on pp. 4, 80).

[31] *International Survey on Private Copying: Law & Practice 2015*. Tech. rep. WIPO and Stichting de Thuiskopie, 2016 (cit. on p. 4).

[32] Johnson, D. and Miller, K. *Computer Ethics: Analyzing Information Technology*. Prentice Hall, 2009 (cit. on p. 3).

[33] Katti, S., Cohen, J., and Katabi, D. "Information Slicing: Anonymity Using Unreliable Overlays". In: *Proceedings of the 4^{th} USENIX Conference on Networked Systems Design and Implementation*. NSDI'07. 2007 (cit. on p. 12).

[34] Khan, S., Aziz, B., Najeeb, S., et al. "Reliability of network simulators and simulation based research". In: *Personal Indoor and Mobile Radio Communications (PIMRC), 2013 IEEE 24^{th} International Symposium on*. Sept. 2013, pp. 180–185 (cit. on pp. 49, 83).

[35] Le Blond, S., Manils, P., Chaabane, A., et al. "One Bad Apple Spoils the Bunch: Exploiting P2P Applications to Trace and Profile Tor Users". In: *Proceedings of the 4^{th} USENIX Conference on Large-scale Exploits and Emergent Threats*. LEET'11. 2011 (cit. on pp. 2, 79).

[36] Liao, Q., Li, Z., and Striegel, A. "Is more P2P always bad for ISPs? An analysis of P2P and ISP business models". In: *23^{rd} International Conference on Computer Communication and Networks (ICCCN)*. Aug. 2014, pp. 1–6 (cit. on pp. 8, 11, 16, 30, 82).

[37] Lin, S. J., Al-Naffouri, T. Y., and Han, Y. S. "FFT Algorithm for Binary Extension Finite Fields and Its Application to Reed-Solomon Codes". In: *IEEE Transactions on Information Theory* 62.10 (Oct. 2016), pp. 5343–5358 (cit. on p. 17).

[38] Lin, S.-J., Chung, W.-H., and Han, Y. S. "Novel Polynomial Basis and Its Application to Reed-Solomon Erasure Codes". In: *Proceedings of the 2014 IEEE 55^{th} Annual Symposium on Foundations of Computer Science*. FOCS '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 316–325 (cit. on p. 17).

[39] Luby, M. "LT codes". In: *Foundations of Computer Science, 2002. Proceedings. The 43^{rd} Annual IEEE Symposium on*. 2002, pp. 271–280 (cit. on p. 16).

[40] Mohaisen, A. and Kim, J. "The Sybil Attacks and Defenses: A Survey". In: *Smart CR* 3.6 (2013), pp. 480–489 (cit. on p. 29).

[41] Nagle, J. *Congestion Control in IP/TCP Internetwork*. RFC 896. Jan. 1984, pp. 1–9 (cit. on p. 68).

[42] Nielson, S. J. and Wallach, D. S. "The BitTorrent Anonymity Marketplace". In: *CoRR* abs/1108.2718 (2011) (cit. on p. 13).

[43] Pelkey, J. and Riley, G. "Distributed Simulation with MPI in ns-3". In: *Proceedings of the 4^{th} International ICST Conference on Simulation Tools and Techniques*. SIMUTools '11. Barcelona, Spain: ICST, 2011, pp. 410–414 (cit. on p. 50).

[44] Petrocco, R., Capotă, M., Pouwelse, J., and Epema, D. H. "Hiding user content interest while preserving P2P performance". In: *Proceedings of the 29^{th} Annual ACM Symposium on Applied Computing*. ACM. 2014, pp. 501–508 (cit. on p. 14).

[45] Poort, J. and Quintais, J. P. "The Levy Runs Dry: A Legal and Economic Analysis of EU Private Copying Levies". In: *JIPITEC - Journal of Intellectual Property, Information Technology and E-Commerce Law* 4.3 (2013), pp. 205–224 (cit. on p. 4).

[46] Pouwelse, J., Garbacki, P., Epema, D., and Sips, H. “The BitTorrent P2P File-sharing System: Measurements and Analysis”. In: *Proceedings of the 4^{th} International Conference on Peer-to-Peer Systems*. IPTPS’05. Ithaca, NY: Springer-Verlag, 2005, pp. 205–216 (cit. on p. 68).

[47] Price, D. *Sizing the piracy universe*. Tech. rep. NetNames, Sept. 2013 (cit. on pp. 2, 79).

[48] Quintais, J. P. “Private Copying and Downloading from Unlawful Sources”. In: *IIC - International Review of Intellectual Property and Competition Law* 46.1 (2015), pp. 66–92 (cit. on p. 4).

[49] Qureshi, J., Heng Foh, C., and Cai, J. “Primer and Recent Developments on Fountain Codes”. In: *Recent Advances in Communications and Networking Technology (Formerly Recent Patents on Telecommunication)* 2.1 (2013), pp. 2–11 (cit. on p. 16).

[50] Reed, I. and Solomon, G. “Polynomial Codes Over Certain Finite Fields”. In: *Journal of the Society for Industrial and Applied Mathematics* 8.2 (1960), pp. 300–304 (cit. on pp. 8, 16).

[51] Reed, I., Truong, T., and Welch, L. “The fast decoding of Reed-Solomon codes using number theoretic transforms”. In: *to The Deep Space Network Progress Report* (1976), pp. 42–35 (cit. on pp. 37, 40, 41).

[52] Riley, G. F., Ammar, M. H., and Fujimoto, R. “Stateless Routing in Network Simulations”. In: *in Proceedings of the Eighth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*. 2000, pp. 524–531 (cit. on p. 20).

[53] Savola, P. “The Ultimate Copyright Shopping Opportunity – Jurisdiction and Choice of Law in Website Blocking Injunctions”. In: *IIC - International Review of Intellectual Property and Competition Law* 45.3 (2014), pp. 287–315 (cit. on p. 4).

[54] Shieh, M.-Z., Tsai, S.-C., and Yang, M.-C. “On the inapproximability of maximum intersection problems”. In: *Information Processing Letters* 112.19 (2012), pp. 723 –727 (cit. on p. 31).

[55] Shojania, H. and Li, B. “Tenor: making coding practical from servers to smartphones”. In: *Proceedings of the international conference on Multimedia*. MM ’10. Firenze, Italy: ACM, 2010, pp. 45–54 (cit. on p. 16).

[56] Shokrollahi, A. “Raptor codes”. In: *Information Theory, IEEE Transactions on* 52.6 (2006), pp. 2551–2567 (cit. on p. 16).

[57] Soro, A. and Lacan, J. “FNT-based Reed-Solomon erasure codes”. In: *Proceedings of the 7^{th} IEEE conference on Consumer communications and networking conference*. CCNC’10. Las Vegas, Nevada, USA: IEEE Press, 2010, pp. 466–470 (cit. on pp. 17, 38, 47, 48, 83).

[58] Spinello, R. A. “Intellectual Property: Legal and Moral Challenges of Online File Sharing”. In: *The Handbook of Information and Computer Ethics*. John Wiley & Sons, Inc., 2009, pp. 553–569 (cit. on pp. 1, 2).

[59] *State of the Internet Q3 2016*. Tech. rep. Akamai, 2016 (cit. on p. 68).

[60] Stutzbach, D. and Rejaie, R. “Understanding Churn in Peer-to-peer Networks”. In: *Proceedings of the 6^{th} ACM SIGCOMM Conference on Internet Measurement*. IMC ’06. Rio de Janeriro, Brazil: ACM, 2006, pp. 189–202 (cit. on p. 67).

[61] Swenson, B. P. and Riley, G. F. “Simulating Large Topologies in Ns-3 Using BRITE and CUDA Driven Global Routing”. In: *Proceedings of the 6^{th} International ICST Conference on Simulation Tools and Techniques*. SimuTools ’13. Cannes, France: ICST, 2013, pp. 159–166 (cit. on p. 50).

[62] Trifa, Z. and Khemakhem, M. “Mitigation of Sybil Attacks in Structured P2P Overlay Networks”. In: *2012 Eighth International Conference on Semantics, Knowledge and Grids*. 2012, pp. 245–248 (cit. on p. 29).

[63] Tsang, P., Kapadia, A., Cornelius, C., and Smith, S. “Nymble: Blocking Misbehaving Users in Anonymizing Networks”. In: *Dependable and Secure Computing, IEEE Transactions on* 8.2 (Mar. 2011), pp. 256–269 (cit. on p. 1).

[64] Vitorino, A. *Recommendations resulting from the mediation on private copying and reprography levies*. Tech. rep. Jan. 2013 (cit. on p. 4).

[65] Weckert, J. and Al-Saggaf, Y. “Regulation and Governance of the Internet”. In: *The Handbook of Information and Computer Ethics*. John Wiley & Sons, Inc., 2009, pp. 473–495 (cit. on p. 3).

[66] Whitman, M. and Mattord, H. *Principles of Information Security*. Cengage Learning, 2014, pp. 89–116 (cit. on pp. 1, 3, 79, 80).

Online

[@1] *Berne Convention*. URL: `http://www.wipo.int/treaties/en/ip/berne/` (visited on July 6, 2017) (cit. on pp. 4, 80).

[@2] Borders, R. W. *World Press Freedom Index 2017*. URL: `https://rsf.org/en/ranking/2017` (visited on July 6, 2017) (cit. on p. 3).

[@3] Harrison, D. *BitTorrent Protocol Specification and Enhancement Proposals*. 2017. URL: `http://www.bittorrent.org/beps/bep_0000.html` (visited on July 6, 2017) (cit. on p. 69).

[@4] NS-3 Consortium. *ns-3 Network Simulator*. URL: `https://www.nsnam.org/` (visited on July 6, 2017) (cit. on pp. 8, 49, 83).

[@5] NS-3 Consortium. *ns-3 Network Simulator - TCP Models (Open Issues)*. URL: `https://www.nsnam.org/wiki/Ns-3.27#TCP_models` (visited on July 6, 2017) (cit. on p. 83).

[@6] The Freenet Project Inc. *FreeNet Project*. URL: `https://freenetproject.org/` (visited on July 6, 2017) (cit. on p. 1).