High-throughput real-time onion networks to protect everyone's privacy

Quentin Dufour

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

Thèse de doctorat de

L'UNIVERSITÉ DE RENNES 1

École Doctorale N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : Informatique

Par

Quentin DUFOUR

High-throughput real-time onion networks to protect everyone's privacy

Thèse présentée et soutenue à Rennes, le 24/02/2021
Unité de recherche : Irisa (UMR 6074)

Rapporteurs avant soutenance :

Ian Goldberg	Professor - University of Waterloo
Rüdiger Kapitza	Professor - TU Braunschweig

Composition du Jury :

Rapporteurs :	Ian Goldberg	Professor - University of Waterloo
	Rüdiger Kapitza	Professor - TU Braunschweig
Examinateur·ices :	Anne-Marie Kermarrec	Professor - EPFL
	Julia Lawall	Directrice de Recherche - Inria Paris
	Isabelle Puaut	Professeure des Universités - Univ Rennes, CNRS, Inria, IRISA
	Alain Tchana	Professeur des Universités - ENS Lyon, LIP
Dir. de thèse :	Yérom-David Bromberg	Professeur des Universités - Univ Rennes, CNRS, Inria, IRISA
Co-dir. de thèse :	Davide Frey	Chargé de Recherche - Univ Rennes, CNRS, Inria, IRISA

Remerciements

Je tiens à remercier tout d'abord Ian Goldberg et Rudiger Kapitza qui ont accepté d'être rapporteurs pour ma thèse ; je suis très reconnaissant pour le temps qu'ils m'accordent. J'aimerais également remercier tous les examinateur·ices : Anne-Marie Kermarrec, Julia Lawall, Isabelle Puaut et Alain Tchana, pour avoir accepté de prendre part à mon jury.

Je souhaite également remercier les membres de l'équipe WIDE qui m'a grandement aidé dans mon travail. Tout d'abord, Virginie Desroches, qui m'a accompagné dans toutes les démarches administratives et m'a aidé à sortir la tête de l'eau quand je ne savais plus qui contacter ! Je pense également aux doctorants passés et présents de notre équipe : Adrien Luxey, Louison Gitzinger, Loïck Bonniot, Alex Auvolat, ainse que tous les autres. Nos débats étaient enflammés, vos conseils précieux. Merci également aux permanents de l'équipe : François Taïani, Erwan Le Merrer, George Giakkoupis et Michel Raynal, aux ingénieurs, post-doc, stagiaires, avec qui j'ai toujours eu des discussions très enrichissantes.

Je souhaite remercier chaleureusement Étienne Rivière pour son accueil à Louvain-la-Neuve et pour sa collaboration précieuse les contributions de cette thèse. En Belgique, j'ai également eu le plaisir de rencontrer Raziel Carvajal Gómez, Paolo Laffranchini et de nombreux autres (post-)doctorants que je remercie particulièrement pour leur accueil et que j'ai grand plaisir à (re)voir en conférence.

Bien entendu, je souhaite remercier tout particulièrement mes encadrants, David Bromberg et Davide Frey, pour leur aide précieuse et sans qui rien de tout cela n'aurait été possible. Ils m'ont suivi, conseillé, aidé, été disponibles, beaucoup appris également, mais aussi soutenu, pendant ces trois années de thèse : je leur en suis très reconnaissant.

Dans un autre registre, je souhaite remercier Ophélie Marcel, tou·tes mes amis et ma famille : ils et elles ont répondu présent pendant ces trois années et m'ont permis de concilier travail et vie personnelle de la meilleure façon qui soit.

Enfin, je souhaite remercier les personnes qui m'ont aidé à travers leurs contributions aux communs, que ce soit via le logiciel libre ou l'accès ouvert aux contenus scientifiques. À ce sujet, j'ai une pensée particulière pour le travail d'Alexandra Elbakyan.

Table of Contents

1 Introduction **8**

1.1 Privacy and Surveillance 9
1.2 Law and Regulation 11
1.3 Privacy Enhancing Technologies 12
1.4 Democratizing Anonymity Networks 15
1.5 Our Contribution 16

2 State of the Art **19**

2.1 Designs 19
2.2 Optimizations 22
2.2.1 More Relays 22
2.2.2 Better Circuits 24
2.2.3 Better Relay Design 25
2.2.4 Better Transport 27
2.2.5 Application Specific Optimizations 29
2.2.6 Conclusion 30
2.3 Group Communication 32
2.4 Security 34
2.5 Conclusion 36

3 Low Latency Communication Over Tor **37**

3.1 VoIP networking requirements 38
3.2 VoIP over Tor: How bad is it? 41
3.2.1 Tor in a nutshell 41
3.2.2 Evaluation of Tor onion routes' QoS 42
3.3 Donar: Enabling VoIP over Tor 46
3.3.1 Link monitoring and selection 47
3.3.2 Scheduling policies 49
3.3.3 Establishing communication 49

3.4 Security . . . 50
3.5 Evaluation . . . 53
3.5.1 Performance & SOTA comparison . . . 53
3.5.2 Microbenchmarks . . . 56
3.6 Conclusion . . . 61

4 High-Throughput Communication Over a New Edge-First Onion Protocol 62
4.1 Introduction . . . 62
4.2 Problem Definition and Goals . . . 63
4.2.1 Functional and performance goals . . . 66
4.2.2 Anonymity goals . . . 67
4.3 Approach . . . 68
4.3.1 Overview . . . 68
4.3.2 Relaying Data In An `IOC` . . . 70
4.3.3 Lifetime of an `IOC` . . . 72
4.3.4 Selecting Relays for Multipath Circuits . . . 74
4.3.5 Reliable Transmission and Flow Control . . . 75
4.4 Security Analysis . . . 76
4.4.1 Attacks . . . 76
4.4.2 End-to-end Anonymity Quantification . . . 77
4.5 Evaluation . . . 80
4.5.1 Building an Edge Volatility Dataset . . . 80
4.5.2 Performance & Comparison . . . 80
4.5.3 Anonymity Sensitivity Analysis . . . 83
4.5.4 Predictors Soundness . . . 85
4.6 Conclusion . . . 86

5 Paving the Way to Decentralized Anonymous Group Communication 87
5.1 Network Coding Background . . . 89
5.2 Contribution . . . 91
5.2.1 System model . . . 91
5.2.2 Solving RLNC challenges . . . 92
5.2.3 CHEPIN . . . 93
5.3 Application to Pulp . . . 95

5.4 Evaluation 97
5.4.1 Experimental setup 99
5.4.2 Configuring the Protocols 99
5.4.3 Bandwidth and delay comparison 101
5.4.4 Adaptiveness optimization 102
5.4.5 Behaviour under network variation 103
5.5 Conclusion 105

6 Future Work **107**
6.1 Onion Routing 107
6.2 Group communication 108
6.3 Long-term goals 109

7 Conclusion **110**

A Résumé en français **112**

Bibliography **117**

Chapter 1

Introduction

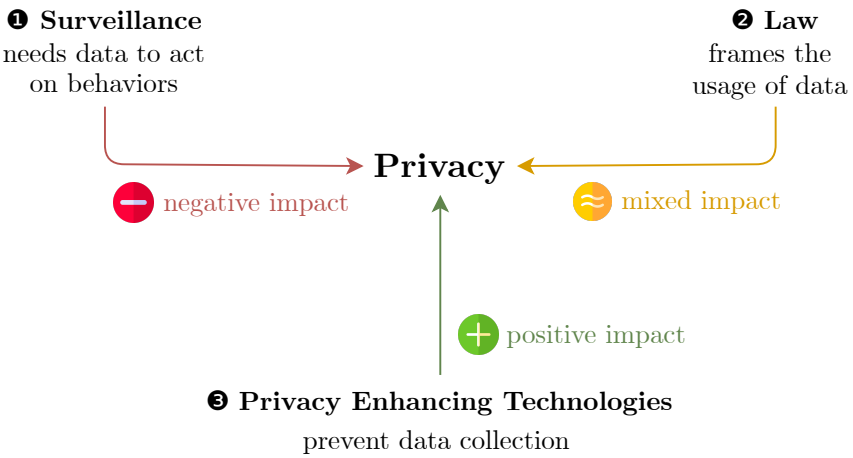

Figure 1.1 – Influence of surveillance, law and privacy enhancing technologies on privacy.

In 2014, Cambridge Analytica silently collected and used personal data to build psychological profiles of millions of Facebook users. Based on these profiles and user data, they served individualized advertisements to their targets to influence votes at the 2016 US elections [1]. While the real impact of this scandal on people's behavior is still not clear [2], we know it is possible to infer the personality of people by only looking at the content they liked on Facebook. Especially, an attacker learning as few as 300 items liked by their target can outsmart people that know this target the best (like their family or friends) when coming to judge their personality [3]. Once revealed in 2018 by the Guardian [4], the Cambridge Analytica scandal generated strong public reactions and made the headlines all around the world [5, 6, 7]. The same year, Facebook's CEO was asked to answer questions in front of the US congress about Facebook's misuse of user data [8, 9]. He declared "We didn't do enough to prevent these tools from being used for harm as well and that goes for [...] data privacy". Far from being an isolated issue, this event seems to be part of a more general and diverse trend of privacy concerns [10, 11,

12, 13, 14, 15, 16, 17] making privacy essential nowadays. To better grasp the common denominators among all these privacy issues, we decided to put these events in perspective with surveillance, law and technologies as summarized in Figure 1.1.

1.1 Privacy and Surveillance

Bennett [18] reviewed many definitions, critiques, and views on privacy before concluding:

"Privacy [...] displays a remarkable resilience as a way to regulate the processing of personal information by public and private organizations, as a way for 'privacy advocates' to resist the excessive monitoring of human behavior. [...] Privacy frames the ways that most ordinary people see the contemporary surveillance issues."

We observed that in the academic world, privacy and surveillance are two tightly linked terms. While "privacy is not the antidote to surveillance" [18], surveillance has strong interactions with privacy and helps to understand its value. Gilliom and Monahan define surveillance as "monitoring people in order to regulate or govern their behavior" [19]. On Figure 1.1 part ❶, we depict the appeal for surveillance as a negative influence on privacy.

Solove [20] proposes metaphors to better grasp the possible negative impacts of surveillance through literary works. According to him, George Orwell's *1984* highlights risks of inhibition and social control of surveillance describing law enforcement's monitoring of citizens. However, the author highlights the fact that surveillance has more pernicious effects better described under a second metaphor: Kafka's *The Trial*. He describes a bureaucracy that collects data about people to make important decisions about them without allowing people to participate in decisions or to even know how they have been taken.

Zuboff [21] frames this second metaphor in light of *big data*, seeing it as "a new logic of accumulation [...] that aims to predict and modify human behavior as a means to produce revenue and market control". In her vision, a bureaucracy is any organization with the material, knowledge, and financial resources to propose communication infrastructures. She defines two subgroups, those who "sell opportunities to influence behavior for-profit" and those who "purchase such opportunities".

Rouvroy and Berns [22] name "gouvernementalité algorithmique" their interpretation of this bureaucracy. Their main criticism is that such data is not used to govern the real

but to govern from the real. Departing from traditional statistics that pose hypotheses and try to prove them wrong or true to take a decision, current data mining practices often consist in inferring rules from a reference dataset. By involving no hypothesis, the authors say that data mining appears more neutral, not affected by any bias but instead, it often reproduces biases in our world without the possibility to know why or to criticize them.

Surveillance creates an environment where entities make decisions about people without them knowing it or how they are taken. Not only does it lead to inhibition and social control, but it also reduces people's agency. [1] Judging that limiting surveillance is beneficial, we explore ways to reduce its impact in the following.

Traces: Surveillance's Fuel There is a consensus [20, 22, 21] that surveillance organizations operate in three main steps. First, they invisibly collect (or extract) data as the service is used. Then they process data, cross them to extract knowledge, and possibly disseminate them to other actors. Finally, they act on behaviors to serve their interests.

One could say that people could not give their data to surveillance organizations, thus preventing data collection and all the following actions. Rouvroy and Berns [22] argue that data is not stolen, as it would enable users to resist collection. Instead, organizations operate a significant weakening of the deliberate nature of information disclosure: data is more abandoned than given. By being trivial, insignificant, segmented, decontextualized, collected data is assimilated to left traces. For a user, it is impossible to imagine or control how these traces will be used. The authors insist on the uselessness of the notion of *personal data* as it is unnecessary to identify individuals to act on them. Instead, it is enough to collect traces, often referred to as *anonymous* data, to use them to predict behaviors.

Zuboff [21] also notes that organizations operating surveillance are regarded by most people as essential for basic social participation. Their service is then provided in exchange for people's information. However, the author argues this deal is unfair. First, surveillance organizations do not face the same constraining regulations, sanctions, or laws as other

1. Agency is the capacity of someone to freely act given its environment, to participate in creating their own behaviour.

professions handling critical data, like attorneys and doctors. Second, surveillance organizations eliminate reciprocity: they know far more about their user population than the user population knows about itself and the surveillance organization.

From these critics, it appears that even the first surveillance mechanism, data collection, is concerning. Indeed, users are forced to abandon their traces to access a service. Their free and informed consent is not guaranteed as they have no practical way to oppose it and no information on how this data will be used. In our example, we can easily see that Facebook is used to maintain relationships with our relatives making it hard to understand how the traces we leave (liking content, checking the news feed, interacting with a friend) will interfere with the content we see on it and the rest of the Internet.

We note that acting at the data collection level would be an efficient way to control surveillance, but we lack a *modus operandi*. In the following, we discuss how regulation impacts data collection, and more specifically, to what extent it can limit it.

1.2 Law and Regulation

As we depicted in Figure 1.1 part ❷, laws only partially address concerns about trace collection and usage. Still, at its roots, lawyers were the first to introduce and define privacy. In 1890, attorneys Warren and Brandeis wrote an essay to advocate for "The Right to Privacy" [23] and present their vision of privacy as "the right to be alone". The most influential definition of privacy in the policy world [18] states that: "the claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others".

Such definitions can lead to very different interpretations: from a restrained one considering only personal and intimate information, to an extensive one considering any behavioral or descriptive information. Current laws, like the GDPR [24], have limited scope as they operate a distinction between *personal data*, like names, phone numbers, addresses, and *non-personal data*. Only *personal data* are protected while *non-personal data*, previously defined as *traces*, are also heavily used for surveillance. In practice, enforcement of existing laws is limited: it also results in *personal data* being used for surveillance in some cases [25, 26].

To improve regulation, Rouvroy and Berns [27] proposed three radical meta-rights to protect privacy: the right to oblivion, the right to disobedience, and the right to have an explanation. However, today no law nor any regulation project captures the spirit of these three meta-rights as defined by the authors. Worse, some of them may be hard to enforce in practice: considering the right to explanation, it is possible to give plausible wrong explanations, similarly to a club bouncer giving untruthful explanations upon customer rejection [28].

We conclude that if some texts exist to regulate the collection and processing of data, they are far from covering the whole surveillance spectrum. Enabling people to efficiently protect themself against surveillance requires other tools: in the following, we explore technical ones.

1.3 Privacy Enhancing Technologies

As the law is lagging behind actual surveillance practices, we promote in this thesis a technical approach known as "Privacy Enhancing Technologies" (PETs) [29, 30, 31]. To avoid most of the surveillance drawbacks, we aim to act upstream, to prevent the creation and collection of traces, as mentioned in Figure 1.1 part ❸. Such an approach has had lots of traction in the recent past, especially with end-to-end encrypted messaging services, as the actual content being exchanged is now increasingly protected using systematic end-to-end encryption; but this is not sufficient. The mere existence of communication between users may reveal sensitive information [11].

Communication metadata indicates who communicated, when, how often, or from which location. It has been shown that accessing metadata without knowing the exchanged content can disclose personal information such as medical conditions, religious beliefs, ongoing legal disputes, financial situations, or political opinions [10, 11]. Due to their nature, communication metadata is critical: network and service providers need to access it to establish communication.

Considered anonymity model If communication metadata cannot be protected by encryption, it can be protected by rendering users anonymous. In this paragraph, we formalize anonymity through 3 properties that are usually considered [32]: *Sender-Receiver Anonymity*, *Sender Unlinkability*, and *Relationship Anonymity*. To define these proper-

ties, we introduce the following notation: S denotes a sender, R a receiver, $\{S \rightarrow R\}$ a communication between S and R, and $\mathcal{A}$ an attacker. Next, we define capabilities that our attacker $\mathcal{A}$ can leverage to threaten our defined properties.

Sender-Receiver Anonymity. If one of the participants is compromised or under surveillance, we want to protect the other participant's identity. In a practical case, this could help journalists under surveillance to protect their sources. If $\mathcal{A}$ knows the receiver R, $\mathcal{A}$ must not be able to determine S's identity. Formally, $\mathcal{A}$ must not be able to distinguish $\{S_1 \rightarrow R\}$ from $\{S_2 \rightarrow R\}$. Conversely, if $\mathcal{A}$ knows the sender S, $\mathcal{A}$ must not be able to determine R's identity. Formally, $\mathcal{A}$ must not be able to distinguish $\{S \rightarrow R_1\}$ from $\{S \rightarrow R_2\}$.

Sender Unlinkability. It must be impossible to determine if two messages come from the same sender or not. Hence, an attacker must not be able to infer communication patterns of participants that could reveal some data about their behavior. Formally, $\mathcal{A}$ must not be able to distinguish $\{S_1 \rightarrow\}$ from $\{S_2 \rightarrow\}$.

Relationship Anonymity. It must be impossible to build a social graph by observing communication, hence two pairs of users communicating must be indistinguishable from one another. Formally, $\mathcal{A}$ must not be able to distinguish $\{S_1 \rightarrow R_1, S_2 \rightarrow R_2\}$ from $\{S_1 \rightarrow R_2, S_2 \rightarrow R_1\}$.

These anonymity properties must hold even under attacks of an attacker $\mathcal{A}$. To conduct its attacks, we consider $\mathcal{A}$ has access to part of the communication infrastructure. $\mathcal{A}$ can be an Internet Service Provider (ISP) that has access to cables and routers. $\mathcal{A}$ can also be a Service Provider serving websites, emails, etc., proposing (malicious) anonymization services. However, we consider that $\mathcal{A}$ has only a partial view of the infrastructure. We quantify the anonymity of the system by establishing a relation between the number of entities that $\mathcal{A}$ controls and the probability that $\mathcal{A}$ can successfully de-anonymize a target.

As $\mathcal{A}$ has access to infrastructures, it can arbitrarily manipulate transferred messages by dropping, replaying, or forging them. In particular, $\mathcal{A}$ may take the role of a corrupt insider to the protocol and attempt to initiate exchanges as if it was a regular user, with the goal of de-anonymizing the communicating partner.

We use this anonymity model for the rest of the document, including our contributions. In the following, we review how well existing Privacy Enhancing Technologies match our anonymity model.

Privacy providers Over the last decades, many solutions have emerged that claim to prevent some intermediaries from collecting communication metadata. We refer to them as *Privacy Providers* as they act as an intermediary between the user and the service. VPN [2] providers declare preventing ISP [3] from spying on their users [33, 34, 35], proxy services circumvent state surveillance (like in China and Iran) [36, 37], whistleblower platforms enable employees to communicate with their organization to report internal frauds [38, 39, 40].

Nevertheless, all these solutions share a common shortcoming: they do not provide any of our anonymity properties against the *Privacy Provider* (VPN provider, proxy service, whistleblower platform). In practice, *Privacy Providers* can be deceptive for users: The Facebook Onavo VPN service was specifically designed to silently collect users metadata and browsing habit while describing itself as "a secure VPN for your personal info" [41]. Even if not deceptive, these actors can be compromised [42]. Once the metadata are in the hands of a deceptive actor, users' communication metadata face the same threats that pushed the users to use a *Privacy Provider* in the first place.

Anonymity networks Contrary to the aforementioned solutions, anonymity networks prevent communication metadata collection without allowing an intermediary to collect these metadata [43]. Over the last 40 years, three major designs with different privacy/performance trade-offs have been explored: mix networks [44, 45, 46, 47, 48, 49, 50, 51], dining cryptographer networks [52, 53, 54, 55, 56], and onion routing [57, 58, 59, 60, 61, 62, 63]. All designs satisfy our anonymity model.

As part of our work, we aim to enforce these three anonymity properties: *sender-receiver anonymity*, *relationship anonymity*, and *sender unlinkability* to protect users' communication metadata from an attacker having a partial view of the network. We note that anonymity networks are far better than *Privacy Provider* at enforcing these properties as they do not trivially enable communication metadata collection. While freely available, we observe that anonymity networks are only used by a small fraction of internet users.

2. Virtual Private Networks
3. Internet Service Providers

1.4 Democratizing Anonymity Networks

While anonymity networks well protect communication metadata, we observed that their adoption by the wide public is limited. One major reason that hinders their adoption is their performances: latency is high, throughput is low [64]. In the following, we explore the different challenges involved to get the best performances for anonymity networks to target a greater adoption.

Global Attacker is too costly We note that anonymity-network designs have different security assumptions. Mix networks and DC networks enforce anonymity against an attacker that can observe all network links, referred to as a Global Passive Attacker (GPA) while onion routing does not. Being resistant to the GPA has a cost, as GPA-proof protocols require either to increase latency by batching messages or to reduce usable throughput by continuously sending. As mentioned in our security model, we do not target a such powerful attacker making onion-routing a viable solution for us.

In practice, end users of DC and MIX networks can expect low throughput, no more than 1 kb/s, and high latencies, around 1 sec or more [46, 47]. Comparatively, onion routing systems provide multiple Mb/s throughput: we measured an average of 5 Mb/s over Tor while having median latencies around 200ms. In conclusion, it seems very hard to use Mix and DC networks to do more than asynchronous text messaging or similar communication.

Onion routing does not require batching messages or sending continuously to provide its security features which explains why it can provide way better performance. In the following, we discuss why despite these optimistic figures, onion routing performances are still too bad to enable a wide adoption.

Application incompatibility We observed that VoIP [4], file-sharing, and group collaboration applications do not work well with Tor, the biggest deployed onion-routing yet. As we will discuss next, these limitations are due to the underlying communication link provided by Tor that is not adapted to considered applications.

If median latencies on Tor, an onion routing system, are in the same order of magnitude as regular internet connections, tail latencies are multiple orders of magnitude higher [64]. This is due to Tor's design that has a complex congestion control and no coordination

4. Voice over Internet Protocol, also called IP telephony

between clients which can lead to temporary congestion on some relays [65]. Such high latencies prevent the use of many real-time applications over Tor like VoIP.

Considering throughput, Tor also features severe limitations. For example, it would not be able to sustain a popular file sharing service such as WeTransfer, which required 120 Gb/s in 2014 [66]. While the software has been developed under the OnionShare umbrella, the Tor network has not enough bandwidth to sustain WeTransfer traffic alone. As of 2020, around 6000 relays were registered in the Tor consensus [5] advertising a raw 500 Gb/s throughput [6]. As OnionShare requires 6 relays to forward data traffic, the raw throughput must be divided by 6 to obtain the usable one, 83 Gb/s, far below that required by a single service (WeTransfer) 6 years ago.

Finally, Tor lacks a way to provide anonymous group communication as it lacks a primitive for it. In practice, people coordinate themselves around existing centralized software exposed behind an onion service like a mail server or forum software. While users and service providers remain anonymous, encryption is often lost from the service provider's point of view. When not lost, at least *sender unlinkability* is. It enables the service provider to spy on its users' behaviors even if it does not know them. We claim that it is desirable to build a group communication primitive over Tor that does not involve a third-party service provider.

In our goal to democratize anonymity networks, we focus our work on onion-routing as a building block as (i) most deceptive actors are not GPA (relatives, employers, companies), (ii) performance is an important factor of adoption, and (iii) we pursue a positive social impact by targeting the right level of anonymity (increasing privacy while preventing impunity). We observed that onion-routing is still suffering from latency and throughput issues that prevent a class of applications (VoIP, file-sharing, group collaboration) from being used. In the following, we discuss how we addressed these limitations.

1.5 Our Contribution

In this thesis, we started by analyzing network requirements of the common applications that are VoIP software, file-sharing, and group collaboration tools. We then asked ourselves how we could make them privacy-friendly by making them compatible with an

5. The Tor consensus is a file containing the list of all the active relays in the network including their IP address, their public key, their estimated bandwidth and some Tor's specific flags.

6. See Tor Metrics at `https://metrics.torproject.org/`.

anonymity network, especially by solving latency and throughput issues. We iterated over an existing design, onion-routing, to see in each case what is the minimal modification that is required to enable the corresponding application.

Reviewing the literature (Chapter 2) convinced us that the design of onion routing can sustain the previously mentioned forms of communication: VoIP, file sharing, and group collaboration. However, existing implementations and optimizations do not provide stable low latencies, enough throughput to sustain a wide adoption, and appropriate group communication primitives. Our contribution focuses on improving these points until making VoIP, file sharing, and group collaboration possible.

Real-time Nowadays, VoIP is unusable over Tor: the sound is so hashed that interlocutor's voice is unintelligible. It appeared it was due to Tor latency variations as latency spikes over 10 seconds are encountered by most Tor circuits. We introduced Donar[7] (Chapter 3), a client-side system that meets VoIP latency requirements over legacy Tor. As a result, we were able to have an easy-to-follow conversation. Under the hood, we maintained a one-way delay below 200ms for 99% of the packets. Our system does not require to send more data on the network.

High-throughput We noted that transferring large files over Tor is discouraged [67] and more generally any throughput intensive applications as Tor network bandwidth is limited. In response, we propose eTor[8] (Chapter 4) as a design shift compared to existing onion networks. We downloaded one year of Tor consensus and observed that already 50% of Tor relays are residential relays, showing the willingness to host relays from home. Our contribution enables Tor users to provision new relays at home, even if volatile and badly-connected, while not impacting the quality of service. With this protocol, it is now possible to encourage users running relays from home with the goal to grow the Tor network. Considering collected residential relays' availability patterns, our new design allows us to make circuits that have an availability close to 100% despite residential-relay churn, compared to only 66% for the current scheme. We show that, as long as 1% of users run a relay, the de-anonymization probability will be inferior to the one on Tor.

7. Donar passed NSDI 2021's first round review but we are still waiting for the final decision.
8. We plan to submit eTor to the USENIX Security 2021 conference.

Group communication Group collaboration over Tor, like forums, emails, and editing tools, is built on top of existing client/server applications. These applications tend to ruin privacy efforts done at the communication levels by Tor by making available to third-parties critical information at the application level. CHEPIN [9] (Chapter 5) is a gossip algorithm to make distributed group communication over onion networks affordable while not exposing group information to a third-party. We show that compared to the state of the art gossip protocol Pulp [69], our system is less sensitive to its configuration parameters. It reduces messages and data redundancy overhead of around 25% while still allowing to reach the whole group. Furthermore, it does not increase latency. We aim to pave the way for anonymous and efficient group communication: with this performance increase, we aim to make gossip affordable over Tor's onion services.

Finally, we discuss how these contributions could spawn new privacy-preserving communication ecosystems (Chapter 6) and conclude (Chapter 7).

9. CHEPIN led to the following publication: Yérom-David Bromberg, Quentin Dufour, and Davide Frey, « Multisource Rumor Spreading with Network Coding », *in*: *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, IEEE, 2019, pp. 2359–2367 [68].

Chapter 2

State of the Art

In our goal to promote strong privacy to the masses *via* Privacy Enhancing Technologies, we started by reviewing network designs that provide anonymity. We focus our analysis on their capability to provide real-time and high throughput communication. We start by reviewing existing anonymity network designs (Section 2.1), we follow by discussing optimizations on onion routing networks (Section 2.2), we study gossip protocols to implement group protocols over anonymity networks (Section 2.3), finally, we review attacks that can be conducted against anonymity networks (Section 2.4).

2.1 Designs

Anonymity networks were introduced by Chaum [43] in 1981 by proposing a way "to hide who a participant communicates with as well as the content of the communication — in spite of an unsecured, underlying telecommunication system". His work gave birth to 3 main derivative designs: mix-net, dc-net, and onion routes that we review in the following.

Mix-net-based networks. Mix networks [44] batch and shuffle packets via *mix nodes* to prevent attackers from performing global traffic analysis. However, in doing so, they inherently incur high latency, which makes them unusable in latency-sensitive applications. A key solution to reduce packet delivery times consists of using cover traffic to prevent the mixes from having to wait too long before having enough packets to send a batch. Accordingly, the challenge faced by the latest research on mix-nets, such as Karaoke [46], Vuvuzela [51], Riffle [45], Loopix [49], Aqua [48], and Stadium [50], consists in designing an adequate mix-net with the best trade-off between minimizing the necessary cover traffic while guaranteeing good resilience to traffic analysis. In their best-case usage scenario, these approaches drastically reduce latency from several hundred seconds to a few seconds, but this remains very far from real-time requirements. Similarly, the high number

of relays involved drastically multiplies the bandwidth usage on the network compared to the payload, limiting in practice the available throughput.

DC-net based networks. Latency can be reduced by avoiding batching. Instead of using mix nodes, Dining-Cryptographer Networks (DC-nets) rely on anonymous broadcast among all network participants [44]. DC-nets have two inherent shortcomings: (i) they incur a high bandwidth overhead, i.e the number of messages exchanged to send one message anonymously grows quadratically with the number of network participants, and (ii) they are vulnerable to denial of service attacks from malicious participants that can jam the whole network. Being resistant to such attacks requires for instance the use of zero-knowledge proofs to detect misbehavior, but this is very costly in terms of computation time and results in increased delivery latency [52]. Consequently, numerous research works on DC-nets have emerged in recent years. Dissent [53, 54], Riposte [55], and Verdict [56] resist jamming attacks while trying to provide the best trade-off between reducing the number of exchanged messages (e.g by splitting the network into smaller parts) and the impact of computational cost on latency. However, despite their efforts, their latency remains far too high for real-time communication. Their design also prevents any bandwidth-intensive communication.

Onion-route-based networks. Departing from the observation that existing applications would not work on top of mixes or dc-net, scholars focused their research on low-latency bidirectional communication networks. It started with the Anonymizer [70], a simple proxy that strips origin information from the communication. However, it requires to trust the server and to be sure that traffic entering and exiting the server can not be observed.

More complex designs based on multiple (non-colluding) servers that know only their predecessor and their successor allow solving the previous problems. The Java Anon Proxy [71] chains multiple proxies in a fixed route named *cascade*: all users take the same route. As a downside, observing the first and last proxy enables to de-anonymize all the users via traffic correlation. Other alternative designs were explored but suffered from structural limitations. PipeNet [72] suffers from trivial denials of service on the whole network. ISDN mixes [73] makes environment assumptions that do not fit the current Internet topology.

The Onion Project [60] introduced the onion route design. It explored the different

challenges involved with deploying an anonymity system: handling anonymous connections [74], configuring and integrating the system into the existing ecosystem [75], and security concerns [76]. In parallel, numerous systems were proposed where the whole infrastructure is contributed by users [59, 77, 78, 79, 80, 81]. Coined peer-to-peer systems, they are now regarded with caution [82] due to recurring attacks, especially on their relay discovery protocols [83, 84, 85].

Finally, the Onion Project [60] froze its final design with Tor [58, 82]: its design mainly consists of fixed-length three-hop virtual circuits with perfect forward secrecy, onion servers to provide receiver anonymity, directory servers to discover peers, and the SOCKS proxy to integrate into the existing ecosystem. Since then, it is the largest deployed anonymity network and the best candidate to enable new services to the masses.

	Latency	Throughput	GPA	References
MixNet	X	∼	✓	[46, 51, 45, 49, 48, 50]
DC-Net	∼	X	✓	[52, 53, 54, 55, 56]
Onion Routing	✓	✓	X	[60, 74, 75, 76, 59, 77, 78, 79, 80, 81, 58, 82]

Table 2.1 – Anonymous network design comparison

In Table 2.1, we compare the three designs through their ability to provide low `Latency`, high `Throughput`, and `GPA`-proof communication. From the three designs, only Onion Routing (OR) ticks both `Latency` and `Throughput` abilities at the cost of being vulnerable to a `GPA`. Such abilities come from OR design: messages are relayed as fast as possible (no re-ordering) with limited overhead (no traffic shaping). We note that even if OR anonymity guarantees are weaker than other designs, they still meet our security model and consequently our privacy goals. By featuring the best performance among anonymity networks while fitting our anonymity model, we focus our following review solely on onion routing.

2.2 Optimizations

In practice, Onion Routing, including Tor, still features network performance issues despite its advantageous design. Indeed, it requires to relay traffic among multiple community-managed relays that can suffer from congestion and downtime. On Tor, these issues are concerning enough to be listed as an open research question [64] and generated many scholarly works to improve it [86, 87]. In this section, we review existing work on Tor and other onion routing solutions in light of our target: enabling new services for the masses involving real time and high throughput communication like VoIP and file transfer. In our review, we start at the network scale, reviewing how we can grow the Tor network (Section 2.2.1), how we can optimize circuits (Section 2.2.2), how to improve relays (Section 2.2.3), how to have more effective transport (Section 2.2.4) and finally what application-specific optimizations we can leverage (Section 2.2.5).

2.2.1 More Relays

One way to improve both latency and throughput is to reduce relays' congestion. The most straightforward approach to reduce congestion on anonymity networks is to increase the number of relays available. In this section, we review known scaling issues and the different ways to encourage the community to run relays.

Directory Scalability As relay discovery appeared to be a sensitive security problem [83] (see Section 2.4), Tor announces all relays to all clients. Such a design leads to a bandwidth increase both when user and relay numbers increases. It leads to forecasts where Tor users would mainly use their bandwidth to download the consensus and not anymore to transfer useful data. To cope with this issue, designs were studied where users learn only a subset of the relays without having an attacker in the system learning what subset the user knows [88, 89, 90, 91]. Such works are required to be able to add new relays to the network but do not encourage people to run them: in the following, we review how incentives could alleviate this problem.

Incentives Many scholars investigated ways to add incentives on the Tor network, to prevent a *Tragedy of Commons* [92] regarding relays bandwidth. First, by encouraging users to send less data by dynamically prioritizing interactive low-bandwidth communication like web browsing over non-interactive high-bandwidth communication like file

sharing [93, 94]. Another design explores how contributing bandwidth to the network as a relay can be directly rewarded with higher priority circuits for the contributor [95, 94]. Finally, some designs introduce a currency to decorrelate resource providers and resource users [96, 97, 98]: users pay to have higher bandwidth and fewer latencies creating an anonymity market. None of these solutions are deployed over Tor as some questions remain unsolved, especially how to fully cope with byzantine relays. Instead, we focus on how to reduce contribution costs by enabling any user to contribute seamlessly.

Reduce Contribution Cost P2P systems, where users also maintain the infrastructure, enable a 0:1 organic scaling: when a user joins, infrastructure capabilities scale accordingly. However, as seen before, onion routing has a poor experience with P2P systems [59, 77, 78, 79, 80, 81] where many security attacks were found but limited to the relay discovery. Considering the directory optimizations discussed earlier, it would be possible to distribute the Tor relay daemon with the Tor client daemon and provide this desirable organic scaling. In practice, there are two main constraints due to residential internet. First, users do not always have full control of their network. As a response, Whisper [63] proposes solutions to enable nodes behind a NAT to open circuits. Second, there remains the open question of the users' devices' churn. Indeed, once a circuit is opened, Whisper or P2P systems do not provide any mechanism to ensure the availability of the circuit across time.

Tor scales well at the relay level but at the directory level the cost is quadratic [88, 91]. Scholars have successfully proposed alternative designs for the directory to drastically reduce its complexity. Apart from the Tor directory scalability issue, there remains the problem to convince the community to run relays, what we refer as *organic scaling*. While many incentives were proposed, it may be hard to ensure relay honesty while it may exclude the most vulnerable people. Exploring the way to reduce the cost of contributing relays seems promising but does not solve all Tor software limitations when it comes to enabling low-latency high-throughput communication. In this section, we discussed how to add more resources to the network but it does not ensure that they will be optimally used. In the next section, we will discuss how to get the most of the available relay network in terms of throughput and latency.

2.2.2 Better Circuits

Over Tor, relays of a circuit are chosen close to randomly by the client, without any coordination. Furthermore, circuits are costly to create as they require many round trips to be opened. We review, in the following, how to open circuits more quickly and better spread loads over the network to reduce congestion.

Circuit Construction First, relays must be configured to learn the existence of the circuit, which takes time and delays the opening of the connection. Indeed, opening circuits has a quadratic complexity in term of exchanged messages over Tor. It is due to its *telescoping mechanism* [58] [1] used to provide perfect forward secrecy. Since then optimizations of circuit constructions have been proposed: Certificateless Onion Routing [99], Pairing Based Onion Routing [100], Diffie-Hellman Optimizations [101, 102, 103]. But in practice, to not penalize user experience, circuits are built in advance: the client daemon keeps a pool of already opened ready-to-use circuits. We conclude that there is no benefit from improving circuit opening performance to pursue our goal (low-latency high-throughput communication) as circuit opening is already invisible to users. It also explains why Tor developers kept their telescoping mechanism.

Path Selection From a circuit perspective, performance is impacted by two factors: queuing (inside the relay) and transmission delays (sending a packet on the "wire", between two Tor relays). By carefully selecting links, scholars were able to reduce transmission delays [104, 105, 106, 107]. However, performance variations are also due to queuing delays [108]. Path selection using relay performance history [109, 110, 111], global coordination [112], probing circuits on their creation [113, 107] all aim to reduce queuing delays. Such predictions have in common to base their choice on a static state in the (close) past. Due to the coarse-grained approach of this selection, it does not capture transient performance variations that drastically harm real-time applications.

We have seen that circuit construction has no direct impact on the final circuit latency or throughput and can be safely kept as it stands. Path selection solutions aim to improve average performance but do not protect from transient performance deterioration.

1. First, the circuit is built only to the first relay resulting in a one-hop circuit. Next, it is extended to the second relay resulting in a two-hop circuit. The final three-hop circuit is achieved by extending the two-hop circuit to the third relay. These successive circuits extensions inspired the *telescoping* term.

Next, we review how forwarding packets can improve performances, even for long-lasting connections.

2.2.3 Better Relay Design

Tor developers chose to use TCP between each relay, resulting in circuits made of multiple chained TCP connections. These original decisions lead to the addition of multiple mechanisms that can harm latency and throughput. Tor's design results in the lack of end-to-end control flow which led to the addition of a window-based control flow by the Tor designers. Moreover, each relay handles hundreds of TCP connections, requiring a strategy to know when and what socket to read or write. Furthermore, Tor multiplexes multiple circuits over a single connection that makes it harder to schedule according to circuits. For example, a cell's circuit is not known before reading it on the socket. In this section, we see how all these design choices negatively impact latency and throughput over Tor and what has been proposed to improve them.

Control Flow Tor relays traffic through circuits made of multiple chained relays connected with point-to-point TCP sockets. As data progresses through relays, data is acknowledged on the socket and can not be dropped anymore as there is no end-to-end retransmission mechanism. If the upstream relay is faster than the downstream one, the queue will indefinitely grow in the upstream relay, leading to relay memory and end-to-end latency increasing towards infinite.

To prevent such a situation from occurring, Tor has two end-to-end control flow mechanisms limiting the number of packets that can be in-flight at the same time (named the window). First, in a circuit, the window contains 1000 cells (512 kB) and an acknowledgment (`SENDME`) is sent every 100 cells. Inside a circuit, a client can send multiple streams on which a control flow is also operated. For each stream, a 500 cell window (256 kB) is kept and an acknowledgment is sent every 50 cells.

These windows are static and relatively large [114] and lead to huge circuit queues inside relays before throttling the sending leading to high latencies in turn. AlSabah et al. [114] studied the problem and proposed two solutions, better configuring the window and introducing a state-of-the-art circuit control flow mechanism working on a per-link basis named N23 [115]. Tuning the window enabled a small decrease in tail latencies but increased download time ; as a result the authors deemed Tor's control flow mechanism as not very effective. Conversely, N23 should be the preferred approach according to the

authors as it enables a more substantial decrease in tail latencies while also reducing download time. However, control flow is not the sole source of latency in a Tor circuit.

Rate Limiting As Tor relay operators may want to only allocate part of their server resources to Tor, they might set a bandwidth limit. The bandwidth limit is enforced in Tor through a token-bucket. When the token-bucket is empty, no more traffic is relayed until the periodic bucket refill occurs.

Historically refilled every second as a tradeoff between liveness and performance, the token has been shown to harm latency, arbitrarily adding 2 seconds delay on some packets [108]. In 2012 a ticket [116] was opened to optimize the token-bucket refill time. Different refill times have been tested and their impact on time to first byte (TTFB) has been evaluated. A slight improvement in TTFB has been observed for 100ms and 10ms compared to 1s. Finally, the Tor community [116, 117] converged to the value of 100ms.

A token-bucket can add at most its refill time in latency at each relay, ie. 100ms now (and 1s previously). Considering a hidden service connection involving 6 relays, it could mean a 600ms (6s previously) penalty. In practice, a latency penalty of a low as 100ms can represent a non-negligible part of the final latency, which in turn harms real-time protocols.

Scheduling Tor handles many buffers: input and output buffers for TCP sockets in the kernel and per-circuit buffers internally. Management of these buffers can significantly impact the time a packet passes queued inside the relay. The original round-robin mechanism is known [65] to be unfair and sub-optimal when multiple circuits share the same TCP connection. To fight the problem of fairness, one must start to define a fairness rule and prioritize traffic following this rule.

The first studied rule is to aim to share the bandwidth, at a given time, as fairly as possible between users. Based on theoretical work from Hahne [118], Tschorsch and Scheuermann [119] propose to enforce max-min fairness (ie. maximizing the bandwidth of the slowest circuit in the network) property by only keeping a single round-robin scheduler at send time.

Sharing bandwidth among users has shown to be unfair: a user that consumes bandwidth for a long time benefits more from the network than a user that will occasionally transfer data. In this context, resource allocation must not be fair at a given time but over time. To satisfy this constraint, researchers have proposed to prioritize circuits according

to an EWMA indicator of their bandwidth usage over time [120] or more advanced machine learning techniques [121]. Other heuristics were proposed: the authors of KIST [122] noted that interactive web traffic is bursty compared to non-interactive one and thus decided to prioritize short bursts of data over constant-rate data.

To make traffic prioritization more efficient, data must spend as little time as possible in the kernel buffers. The authors of KIST [122] take kernel-informed decisions and send data to kernel buffers only if they know the data can be sent and thus not blocked in a kernel buffer. KIST was integrated into Tor in 2018 [123].

Two goals were pursued when optimizing relays: prioritization, to better share available resources among users, and latency minimization, due to rate limiting and buffers. Still, it seems some weaknesses are due to some design decisions. Moreover, some latency or throughput variations may be external to the relay, like a noisy application running on the same server as the relay. In the following, we study how *transport* changes in Tor could help to cope with its environment.

2.2.4 Better Transport

The Tor developers chose to expose a stream abstraction to the end-user, compatible with most of the existing protocols (eg. HTTP), providing in order, no drop delivery to the destination supported by point-to-point TCP connections between relays. As we have seen in the previous section, this comes at a cost as we need to handle a lot of edge cases in a sub-optimal way.

Single Path Tor transport suffers from design problems [124] that artificially increase latency or reduce bandwidth, especially on high load. One problem is that multiple Tor circuits may share the same TCP connection between two relays resulting in suboptimal performances. A stream losing packets or sending too much data will respectively trigger re-ordering (thus head-of-line blocking) or throttling on the TCP connection, affecting unfairly all other streams sharing the same connection. A higher level problem resides in the fact that TCP provides a stream abstraction, hiding loss and delivering packets in-order at the cost of possible delays referred to as "head-of-line blocking". Such abstraction is not necessarily the best according to the usage: VoIP supports loss but does not tolerate well latency spikes.

One observation made by scholars is that non-interactive transfers, like file transfers, impede interactive ones, like VoIP. As a solution, Torchestra [125] proposes to open two TCP links between each relay: one for non-interactive traffic, one for interactive one. TCP-over-DTLS [126] and PCTCP [127] are a generalization of this behavior: the congestion control is done on a per-stream basis and thus done independently of the encrypted link between the two relays. iMUX [128] proposes an improvement on Torchestra [125] while pointing out that opening one socket per-stream [126, 127] is vulnerable to socket exhaustion attacks.

Tor and previous solutions still provide point-to-point congestion control that can slow-down traffic. Tor's socket abstraction does not require that packets are sent ordered between each relay, only between the sender and the recipient. Following this observation, UDP-OR [129] proposes to forward packets on UDP transport between relays and operate a TCP sockets only end-to-end. Pushing the idea further, scholars proposed an end-to-end abstraction change to better fit different communication needs: UDP [130], unordered TCP [131], and QUIC [132].

None of these modifications have been integrated into Tor as they would require heavy modifications to the software with hard to predict effects.

Multipath Having a multipath approach to the Tor network opens new perspectives to leverage available relays. Recognizing its usefulness for available and real-time communication, MPTCP is being standardized as an RFC [133]. Some works focused on tuning MPTCP specifically for low latencies: by duplicating data on two links [134] or by probing links regularly and scheduling traffic on the fastest one [135]. However, such independent solutions fail to grasp Tor's distinctive features: generic algorithms do not take into account Tor scheduling logic seen earlier leading to sub-optimal choices.

In response, scholars designed multipath protocols tailored for onion routing networks. Originally onion routing did not require circuits as all routing information was stored in every cell. MORE [136] builds on this legacy and routes each cell independently. Such design has an important performance impact, requiring relays to run costly cryptography for each cell while raising numerous security issues (traffic correlation, denial of service, etc.).

In a goal to leverage existing work on Tor anonymity and security, incremental modifications such as MPTCP Tor [137], Conflux [138], mTor [139], and mUDP-OR [140] were proposed. MPTCP Tor, Conflux, and mTor simply aggregate existing Tor circuits,

they differ by the metrics they observe: round trip time and/or window size and the proposed scheduling algorithm based on these metrics. mUDP-OR is based on UDP-OR [129] transport and has two simple scheduling mechanisms: random and round-robin.

All these designs remain relatively generic and do not take into account Tor specificities like scheduling or padding. Consequently, such protocols send more data on the network than needed and their latency remains too high for VoIP. We argue that there are still many multipath designs to explore, from integrating a probing mechanism compatible with Tor-specific scheduling to leveraging the fact that Tor relays make a fully connected graph.

Changing Tor's transport protocol is recognized as being a difficult task by Tor developers [141], requiring the re-design of many components of the current Tor solution, as a consequence, it is considered with prudence. Contrary to single path, some aforementioned multipath approaches could be implemented over legacy Tor and could improve either latency or throughput. In the following section, we analyze how application-level streams, specifically VoIP and file transfers, can also be optimized for performance knowing Tor's underlying components.

2.2.5 Application Specific Optimizations

VoIP and file transfers have different requirements than web transfer, for which Tor has been optimized. VoIP sends small packets and tolerates some loss but requires a stable low latency. Conversely, file transfer requires high throughput but is not affected by latency and does not require strict ordering either. In this section, we review anonymity networks designed to support VoIP and file transfer.

VoIP As seen earlier, solutions based on the mix-net principle do not provide a satisfying user experience. Some works [62, 47] were dedicated to enable VoIP over mix networks. In the following, we introduce their design and explain why they fail to meet industry requirements.

Herd's [62] hybrid approach uses mix nodes along with super peers organized in trust zones. Herd can provide VoIP on its anonymity network with good resistance against global adversaries. Its evaluation shows an expected latency of 400 ms in optimal conditions. The recent work on Yodel [47] removes the concept of trust zones and supports

higher percentages of dishonest nodes than Herd. However, this comes at the cost of latency increasing with the probability of having dishonest mix nodes. For instance, with Tor-like security guarantees (i.e. around 20% of malicious servers) latency already reaches 900 ms. To counterbalance this latency, Yodel uses a codec with poorer quality than industry-standard OPUS.

Even if both Herd and Yodel are promising designs, they were only deployed in controlled and dedicated environments. Today's people are, therefore, unable to communicate using these systems as they have latency superior or equal to the upper bound recommended by the ITU G.114 [142]. Moreover, we point out that the evaluation of both systems has been performed in optimal conditions, and their performance in settings comparable to Tor deployment remains unstudied. For instance, Yodel is evaluated on 100 powerful Amazon EC2 servers with no external interference.

Fakis, Karopoulos, and Kambourakis [144, 143] explore the porting of SIP infrastructures on Tor. The main principle of their work is to preserve privacy in the SIP signaling protocol but it does not leverage Tor built-in mechanisms for signaling, like Onion Services. Moreover, the RTP stream is transmitted using a single Tor onion link: no data-plane improvement is proposed. TorFone [145] improves latency by duplicating traffic over two onion links similarly to ReMP [134]. We demonstrate in Chapter 3 that duplicating data on two links is not sufficient to meet VoIP requirements.

File Transfer A significant share of Tor bandwidth is used to transfer files through BitTorrent [65]. However, it benefits only a minor share of users: the Tor infrastructure can't afford bandwidth-intensive usage. Additionally, BitTorrent does not integrate well with Tor and leaks identities [146].

As an alternative, OnionShare [147] integrates well with Tor and preserves anonymity properties. While it does not leak data, a wider adoption would lead to the collapse of the Tor network. None of these software integrate solutions to scale the network seamlessly with the number of users, similarly to P2P networks. We refer to this property as *organic scaling* and we argue that without it, it will be very hard to maintain enough relays to support bandwidth intensive usage.

2.2.6 Conclusion

In our summary Table 2.2, we built a comparison matrix by judging all our referenced optimizations along four properties: `Signaling`, `Latency`, and `Throughput` improvement

	Signaling	Latency	Throughput	Deployability	References
More Relays					
Directory	✓	X	X	✓	[88, 89, 90, 91]
Incentive	X	✓	✓	XX	[93, 94, 95, 94, 96, 97, 98]
Cost	X	✓✓	✓✓	X	[59, 77, 78, 79, 80, 81, 63]
Better Circuits					
Construction	∼	X	X	∼	[99, 100, 101, 102, 103]
Selection	X	∼	∼	✓	[104, 105, 106, 107, 109, 110, 111, 112, 113]
Better Relays					
Control flow	X	✓	X	∼	[114]
Rate limit	X	✓	X	✓	[108, 116, 117]
Scheduling	X	✓	∼	✓	[119, 120, 121, 122, 123]
Better Transport					
Single path	X	✓✓	X	XX	[125, 126, 127, 128, 125, 129, 130, 131, 132]
Multipath	X	✓✓	✓	∼	[133, 134, 135, 136, 137, 138, 139, 140]
Applications					
VoIP	X	∼	X	∼	[62, 47, 144, 143, 145]
File transfer	X	X	X	✓	[147, 65, 146]

Table 2.2 – Onion routing optimizations comparison

plus `Deployability` easiness.

We identified no possibilities to provide our target real-time and high-throughput properties by acting at the *Circuit* and *Application* level.

Better Relays seems to be desirable as there are many opportunities to improve `Latency` while having a good record on `Deployability`. Unfortunately, many works have already been conducted on this point and possible optimizations are limited by the current transport design and do not take into account the exterior environment of the relay.

Adding *More Relays* is particularly promising as it is the most efficient way to increase `Throughput`. Despite numerous works, we think the incentive approach is not optimal: it would exclude some users due to the cost and misses guarantee on the promised service. Instead, we observe that conditions to run a relay are very strict and hard to ensure at home with personal devices. Through our observations of the Tor consensus, we argue that adapting the protocol to encourage poorly connected devices to participate in the

network could drastically increase its size and result in both a `Latency` and `Throughput` improvement.

Another approach we found promising is to build *Better Transports*. Alternative single path transports over Tor were studied but suffer from three drawbacks: (i) they do not enable a better load balancing on the network, (ii) they provide no redundancy, and (iii) they can't be built on top of the existing network. Comparatively, multipath suffers from none of these limitations: data can be dynamically load-balanced around links improving `Latency` and `Throughput`, multiple links provide redundancy and multipath can in some cases be built on top of the existing network easing `Deployability`.

In the following, we study how we can go from point-to-point communication to group communication over Tor while not involving a central server.

2.3 Group Communication

Often group communication is simply built with a central server multiplexing all connections of the group members. Such solutions are far from being satisfying in terms of privacy, as it introduces an asymmetry between the users and the server. As a solution, epidemic protocols place all actors on an equal footing. First introduced in 1988 by Xerox researchers on replicated databases [148], they can be generalized to any group communication. They introduced three protocols to exchange rumors: push, pull, and push-pull.

Push protocols To transmit rumors, push-based protocols imply that informed nodes relay the message to their neighbors. Some protocols are active, as they have a background thread that regularly retransmits received rumors, like balls and bins [149]. Other protocols adopt a reactive approach, where rumors are directly forwarded to the node's neighbors upon reception, like infect-and-die and infect-forever protocols [150]. Push protocols are particularly efficient to quickly reach most of the network, however reaching all the nodes takes more time and involves significant redundancy, and thus bandwidth consumption.

Pull protocols Nodes that miss messages ask other nodes for the missing messages. As a consequence, *pull protocols* more efficiently reach the last nodes of the network, as inherently, they get messages with higher probability. However, they require sending more messages over the network: (i) one to ask for a missing message, and (ii) another one for

the reply that contains the missing message. Furthermore, a mechanism or a rule is needed to know what are the missing messages to pull, which explains why these protocols are generally used in conjunction with a push phase. Chainsaw [151] uses a pull protocol to learn new data in a dynamic network. Coolstreaming [152] uses a pull protocol to fetch data where the data location was previously learned through a push phase.

Push-Pull protocols The aim is to conciliate the best from push and pull protocols by reaching as many nodes as possible with minimal redundancy on the push phase. Then, nodes that have not received a message will send pull requests to other nodes in the network. By ordering messages, Interleave [153] proposes a solution to discover the missing messages in the pull phase but works only with a single source. Instead of ordering messages, Pulp [69] piggybacks a list of recently received message identifiers in every sent message, allowing multiple sources.

Gossip-based disseminations are characterized by the reception of many redundant messages in the push and pull phases to receive every message with high probability.

Random Linear Network Coding To improve dissemination, some protocols use erasure coding [154, 153] or Random Linear Network Coding [155] but need encoding at the source or message ordering, which limits these techniques to single-source scenarios. Theoretical bounds have also been studied for multi-sender scenarios [156, 157] but they do not consider generations. Generations consist of grouping messages to prevent decoding complexity from exploding, and suppose that messages are previously ordered: they are required for real-world implementations. Network Coding is also used on wireless networks [158, 159] at the physical layer. The setup is different as each message will be received by every node within range.

Applying RLNC gossip in a multi-sender scenario implies determining to which generation a message will belong without additional coordination, and finding a way to link network-coding coefficients to their respective original messages inside a generation.

In our summary Table 2.3, we compare the four aforementioned protocols to `Latency`, `Throughput`, and `Applicability`. We define `Applicability` as how well a considered algorithm integrates to real systems and may answer real needs. No realistic multi-sender network coding (RLNC) epidemic protocol has been proposed yet while it is the most promising in terms of performance. Our goal is to make RLNC algorithms usable in the

	Latency	Throughput	Applicability	References
Push	✓	X	✓	[149, 150]
Pull	X	✓	✓	[151, 152]
Push/Pull	∼	∼	✓	[153, 69]
RLNC	✓	✓	X	[154, 153, 155, 156, 157, 158, 159]

Table 2.3 – Gossip designs comparison

same situations as Push and/or Pull algorithms and therefore tick the `Applicability` property. Once gossip is cheap enough, we can consider running it on top of Tor's onion service.

In the next section, we depart from performance to discuss security: optimizations must not be introduced at the expense of anonymity.

2.4 Security

The optimizations discussed in the previous section could come in conflict with security. In this section, we start by introducing a terminology to define the various desired privacy property to keep the user safe. Camenisch and Lysyanskaya [160] introduced concepts like anonymity, unlinkability, undetectability, unobservability, etc. Such terminology has been revisited [32] to conduct formal analysis on anonymity networks and especially Tor, leading to the security properties presented in Section 1. Based on these properties (unlikability, sender-receiver, and relationship anonymity), we review attacks against them in the following.

Correlation Attacks It is well established that onion routing and Tor in particular are not resilient to end-to-end traffic correlation attacks [58, 161, 162, 163]. An attacker listening to each end of an onion route (by owning both end relays or observing traffic) can easily link sender and receiver, and thus de-anonymize the connection. Such correlation attacks can be based on different traffic observations: by timing packets [161] but also by making an intersection between Tor users set and some exterior sets [164]. In the long run, with relays rotation, only 2 malicious relays in the whole network are needed

to compromise users' anonymity [165, 166]. To cap the cumulative risk of end-to-end correlation, Tor developers made part of the circuit static: the user will pick one relay that will be used as the first hop of all circuits for 4 months. This relay is referred to as an entry guard [167].

Fingerprinting Attacks Fingerprinting works by observing various metrics that are characterizing a user, a machine, or software in a network. Observing latency variations on a Tor circuit makes it possible to identify its bottleneck relay and thus, possibly its guard [168]. Similarly, observing traffic patterns (number of packets, the timing of requests) enables one to identify the public web page that is loaded among a signature database [169, 170]. Given that the attacker is between the user and the guard, it is able to break relationship anonymity by knowing the user and the site they browse.

Onion Services add receiver anonymity to Tor communication, a property that can also be attacked. First, by observing latencies, it is possible for an attacker to determine if a hidden service or a public website is accessed by a user [171]: such information can be used later when conducting a website fingerprinting attack [172]. By connecting to an Onion Service, it is also possible to infer its guard by generating a traffic signature [173]. Some of these attacks can be mitigated through Onion Services version 3 [174], especially by making it possible to publish a private encrypted descriptor.

Fingerprinting leverages knowledge on protocols and networks to de-anonymize some elements of the network. But fingerprinting is not the only attack that leverages knowledge.

Epistemic Attacks Epistemic attacks are based on the fact that an attacker knows that a user only knows a subset of the relays in the network, thus inferring if a given circuit has been possibly or not opened by this targeted user. Such an attack is possible on networks that only advertise a subset N_{subset} of all network relays N_{network}. This need comes from the need to reduce discovery costs discussed in Section 2.2.1. However, doing it naively makes users vulnerable to epistemic attacks. These attacks can drastically reduce the security provided by the network and even lead to de-anonymization when the attacker learns N_{subset} and learns some nodes of the circuit (via a partial network observation or by being part of the circuit). Partitioning the network in $\frac{N_{\text{network}}}{N_{\text{subset}}}$ smaller networks has a negative impact on anonymity as the evaluation must be done on N_{subset} instead of N_{network}.

Letting users pick their own N_{subset} has been shown to be worse in terms of security than statically partitioning the network [175, 83]. Such results particularly discourage peer to peer relay discovery. Since then such a class of attacks has been well studied, as discussed in Section 2.2.1. Still, preventing knowledge-based attacks is not enough: an observer having the capability to observe the network under certain conditions can still de-anonymize users without additional knowledge.

We reviewed three classes of attacks (fingerprint, epistemic, and correlation) that will help us to evaluate the security of our contributions. Fingerprinting and correlation attacks impact transferred data while the epistemic ones impact only circuit construction. We focus our optimization on improving the data transfer and not circuit construction. We still consider attacks on circuit construction as our work could have negative side-effects on it, as seen with the directory scalability issue in Section 2.2.1. We observed that fingerprinting and correlation attacks are mainly considered within the interactive web context. In our contributions, we will reconsider these attacks in light of our targets: VoIP and file transfer. We plan to use Tor as a comparison point for our contributions.

2.5 Conclusion

As Tor is the most widely deployed anonymity networks, each part of its system has been analyzed and is the subject of proposed enhancements. Such enhancements must always be considered under the spectrum of their anonymity impact. Following our goal to enable new usage for Tor: low-latency applications like VoIP and high-throughput like file transfer, we conclude that our goals could be achieved by exploring two promising approaches: reducing contribution cost and multipath. In the next chapter, we show how we were able to propose VoIP over legacy Tor infrastructure with good quality of experience.

Chapter 3

Low Latency Communication Over Tor

We are interested in providing VoIP support over a *readily-available* anonymization network. More specifically, we target a deployment using (1) legacy VoIP applications and (2) the existing, unmodified Tor network. We do not wish to propose design changes to Tor, or a novel anonymity network [49, 50, 51, 62, 47]. We believe that these lines of work are, in fact, orthogonal to our own.

We revisit the assumption that Tor cannot support VoIP. While our observation of the performance of Tor onion links (as presented in **Section 3.2**) confirms that a *single* link cannot provide the stable and low latencies required by high-QoE VoIP. It also allows us to make a case for using *multiple* Tor onion links simultaneously. Our motivation is that the use of multiple onion links, together with controlled content redundancy across them, can mask the transient faults and latency spikes experienced by individual links.

We present the design and implementation of Donar, a user-side proxy interfacing a legacy VoIP application to the existing Tor network (**Section 3.3**).

Donar enforces *diversity* in the paths used for transmitting VoIP packets, i.e., the use of distinct Tor onion links. In addition, we leverage *redundancy* by sending the same VoIP packet several times, using different links. This redundancy does not, in fact, add additional bandwidth costs for the Tor network beyond those incurred by the setup and maintenance of these multiple links. We leverage, indeed, the fact that Tor only transmits 512-Byte cells over the network, in order to protect users against traffic analysis [176, 177]. Donar takes advantage of the available padding space to re-transmit previous VoIP packets. Diversity and redundancy mask the impact of the head-of-line blocking implied by the TCP semantics of Tor onion routes, whereby an entire stream of packets may get delayed by a single belated one.

Donar builds on the following key contributions:

— The *piggybacking* of VoIP packets in the padding space of Tor cells enables redundancy without incurring additional bandwidth costs on the Tor network.
— A *link monitoring* mechanism observes and selects appropriate links, allowing to switch rapidly between links when detecting performance degradation.
— Two *scheduling* strategies for selecting links when transmitting VoIP packets enable different tradeoffs between cost and robustness.

We further analyze in **Section 3.4** how attacks on Tor can affect the security properties of DONAR. In particular, we discuss how different DONAR configurations implement different tradeoffs between Quality-of-Experience and security.

We evaluate DONAR over the Tor network and present our findings in **Section ??**. We use VoIP traffic emulation as well as the off-the-shelf `gstreamer` [178] VoIP client using the OPUS [179] audio codec. We assess the performance of DONAR against VoIP requirements detailed in Section 3.1, and compare it with the approach followed by TorFone [145], a previous design for VoIP over Tor that systematically replicates all packets over two onion links. Our results show that DONAR, using alternatively 6 out of 12 carefully monitored and dynamically selected onion links, achieves latencies under 250 ms with less than 1% of VoIP frame loss for the entire durations of a large number of 90-minute calls, while incurring no bandwidth overhead compared to using a single link in its default configuration.

3.1 VoIP networking requirements

DONAR aims at Providing a good Quality-of-Experience (QoE) for anonymous VoIP while limiting the costs imposed on the Tor infrastructure. We base our analysis of QoE requirements on recommendations by the International Telecommunication Union (ITU) [180, 142, 181]. The ITU defines a good QoE as the combination of the following guarantees: (1) uninterrupted calls, (2) good voice quality, and (3) support for interactive conversations. We analyze in the following these requirements and derive our network QoS objectives, summarized in Table 3.1.

VoIP protocols. VoIP requires two types of protocols. A signaling protocol such as the Session Initiation Protocol (SIP) [182] makes it possible to locate a correspondent and negotiate parameters for the communication. The signaling protocol only impacts QoE with delays upon the establishment of the call. When the call is established, a protocol such as the UDP-based Real-time Transport Protocol (RTP) [183] is used to transmit VoIP

Metric	Objective
Dropped calls rate	$\leq 2\%$ for 90-minute calls
Packet loss rate	$\leq 1\%$
Bandwidth	≥ 32 kbps (4.3 kB/s)
One way delay (99^{th} perc. *ideal*)	≤ 150 ms - T_{frame} - T_{buffer}
One Way Delay (99^{th} perc. *max*)	≤ 400 ms - T_{frame} - T_{buffer}

Table 3.1 – VoIP network performance requirements, following the recommendations of the International Telecommunication Union (ITU) [142] and applying them to the OPUS codec [184, 185].

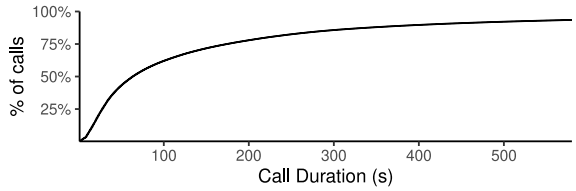

Figure 3.1 – Call Duration ECDF on a 4M calls dataset communicated by Holub et al [**holub2018analysis**] zoomed on first 10 min.

audio frames encoded using a codec, whose configuration is negotiated by the signaling protocol. QoE is primarily impacted by this codec and its ability to deal with hazards in network QoS, as we detail next.

Impact and choice of the audio codec. Bandwidth, latency or maximum packet loss requirements depend on the audio codec used by the VoIP application. We base our analysis on the state-of-the-art open audio codec OPUS, which we also use in our evaluations. OPUS is a widely-used, loss-tolerant audio codec developed by the Xiph.Org Foundation and standardized by the IETF [184, 185]. It targets interactive, low-delay communications and computational efficiency. OPUS has been consistently ranked in comparative studies as the highest-quality audio format for low and medium bit-rates [186, 187]. We emphasize that our analysis would be similar for other open codecs, e.g. the Internet Low Bit Rate Codec (iLBC) [188] or Xiph.Org Foundation's former codecs Vorbis [189] and Speex [190].

First guarantee: *no call interruption.* A call interruption is the most impacting event on user-perceived QoE. The ITU does not provide a recommendation for general networks, but recommends at most 2% dropped calls for VoIP over 4G [181]. We adopt the same goal but need to define a timespan on which to evaluate this metric. Holub

and al [**holub2018analysis**] communicated us a dataset of more than 4M call durations (Figure 3.1). They confirm it follows a log-normal distribution considered as standard for voice calls. We observe an average call duration of a bit above 3 minutes, with 90% of calls lasting less than 10 minutes. However, we still observe 1040 calls lasting for 90 minutes or more which is characteristic of the long tail of the distribution. As it matches major representative carriers limitations [191, 192], we use 90 minute calls as the maximum of a call and thus evaluate reliability over this duration.

Second guarantee: *good voice quality.* Users want to clearly hear their communication partner. Voice quality depends both on the bitrate used and the amount of packet loss:

- Listening tests with OPUS [186, 179] concluded that a bitrate of 32 kbps is sufficient to offer a sound quality that test users cannot distinguish from a reference unencoded version of the recording. We set, therefore, this bitrate as the minimum required link capacity that we must offer to the VoIP application.
- OPUS provides two mechanisms to mask the impact of lost packets: a domain specific one, named Packet Loss Concealment (PLC) and a generic one, via redundancy, named Forward Erasure Coding (FEC) [1] [193]. Han *et al.* [194] studied the perceived quality of a call on various packet rates. This study shows that while PLC compensates for packet loss, the perceived voice quality nonetheless decreases quickly: a 1% packet loss is essentially unnoticed, while 10% packet loss results in usable but degraded call conditions. Based on these results, we set as a requirement a packet loss of at most 1%.

Third guarantee: *interactive conversations.* In addition to an uninterrupted and good-quality voice signal, users of voice calls expect to be able to exchange information interactively, e.g., be able to seamlessly synchronize on when to stop and start talking in a conversation.

Interactivity primarily depends on latency [195]. The ITU published recommendation G.114 [142] on mouth-to-ear latency in voice calls. This recommendation indicates that a delay below 150 ms is unnoticeable for users, compared to a direct voice conversation. We set, therefore, this value as our ideal latency. On the other hand, the recommendation stipulates that delays must remain below 400 ms to make an interactive call possible under good conditions. Higher latencies result in synchronization difficulties and significantly reduce user-perceived QoE. We set this threshold of 400 ms as our maximum acceptable

1. We configure OPUS to use only the former, as DONAR already enables redundancy mechanisms that are specific to the Tor network.

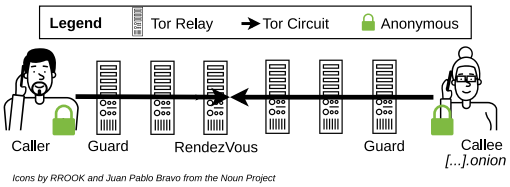

Figure 3.2 – Structure of a Tor link with onion services.

mouth-to-ear latency.

We emphasize that the actual network latency for transmitting VoIP frames is only a subset of mouth-to-ear latency. Additional latency is introduced by (1) audio capture and playing, (2) packetization, and (3) buffering. Pipewire, the default audio framework in Fedora, can enforce delays below 2 ms [**tayman_pipewire_nodate**] on audio capture and playing. We argue this delay is negligible and define as orthogonal to our work possible needed improvements on other platforms. Once digitized, audio is encapsulated in frames every T_{frame} ms that will form a packet. OPUS enables configurable values for T_{frame} from 2.5 to 60 ms.

We consider an ideal jitter buffer model similar to the one from Moon et al [196]. In this model, all frames are delayed to the maximum or n^{th} percentile of observed latency and we allow frame drops. Moon et al [196] and others [197, 198] have proposed jitter buffer implementations performing close to this theoretical optimum. Therefore, we consider T_{buffer}, the unnecessary delay added by a wrong jitter buffer configuration as negligible. Finally, as we allow a 1% frame drop we consider the 99^{th} latency for our mouth to ear delay constraints.

3.2 VoIP over Tor: How bad is it?

In this section, we give a brief overview of the Tor anonymization network (§3.2.1) and report on our own evaluation of its network QoS (§3.2.2) in light of our requirements.

3.2.1 Tor in a nutshell

Tor [199] is a large-scale network that enables users to access remote resources without revealing their identity. Tor relies on *onion routing*: it relays traffic through *circuits* consisting of at least two relays (three by default) chosen from more than 6,000 dedicated nodes. The first relay in a circuit is known as the *Guard*. The Tor client chooses a small

set of n (by default [2], $n = 2$) possible guards. Thereafter, it builds circuits by using one guard from this set, choosing the remaining relays randomly from the list of all available relay nodes.

Tor enables both connections to the regular internet (referred as *Exit*) and to other Tor users (referred as *Onion Services*). Compared to *Exit* mode, *Onion Services* provides two way anonymity between the two participants. Figure 3.2 illustrates an onion service used for transmitting VoIP frames. The caller Tor user connects to an anonymous onion service (the callee) by means of a Tor route, consisting of two Tor circuits, one from the caller to a rendezvous relay, and another from the callee to the same rendezvous relay.

Tor seeks to prevent adversaries from inferring communicating parties. To this end, at least one relay in the onion route should lie in an administrative domain that the adversary cannot observe. Furthermore, to prevent traffic analysis attacks, Tor only sends fixed-sized messages between relays, in the form of 514-Byte *cells* [176, 177]. When a packet being transmitted over a Tor connection is less than 514 Bytes in size, the Tor client pads it with random data to fill the gap.

3.2.2 Evaluation of Tor onion routes' QoS

Tor is often described as a *low-latency* anonymization network. Its TCP streams over pre-established onion routes enable, indeed, lower latency than anonymization networks where the relays for each message in a stream are chosen independently [201, 51, 50]. The latency of onion routes in Tor, and in particular its stability, is however known to be unpredictable, which made several authors doubt of Tor's ability to support low-latency applications such as VoIP [202, 203].

In this section, we report on our own experimental evaluation of the network QoS of Tor onion routes. We confirm the observation made by other authors that a single Tor link is unsuitable for VoIP networking requirements as defined in the previous section. These measurements allow, however, to make the case for the foundational design choice in DONAR: using several dynamically selected links.

We consider the following metrics: connection stability, the variability of one-way latency, and the predictability of high latency from prior measurements. We use a load injector with varying packet-sending rates and, in order to measure one-way latency, a stub communication endpoint located on the same machine. The injector and the stub

2. While Tor advertises using $n = 1$ by default, it effectively uses $n = 2$ [200].

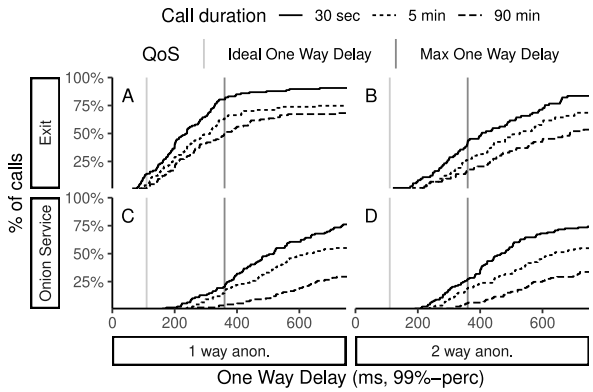

Figure 3.3 – Evolution of the one-way delay's 99^{th} percentile over Tor according to the connection link and the call duration.

use two separate instances of the Tor client in its default configuration, and create circuits independently. All reported experiments were conducted in January 2021.

Connection links. We started by analyzing how the two Tor modes, *Exit* and *Onion Services*, are performing in term of tail latencies. Each of this mode can be declined in one-way and two-way anonymity. *Exit* provides one-way anonymity by default but we can mimic two-way anonymity by making both caller and callee access the same public VoIP server through an *Exit* link. *Onion Services* provides two-way anonymity by default but we can reduce the number of relay and keep only one-way anonymity. We use the HiddenServiceSingleHopMode feature in the Tor daemon to achieve one-way anonymity over Onion Services.

Considering these 4 configurations, we simulated VoIP calls lasting 30 seconds, 5 minutes and 90 minutes. The simulation strictly follow the requirements presented in **Section 3.1**. For each combinations of configuration and call duration, we did 64 calls and present the results in Figure 3.3.

We start our analysis by focusing on Figure 3.3.A as it features the configuration on which Piyush Kumar et al [204]'s made their main claim. With 37% of unacceptable calls (resp. 50%) for 5 minutes (resp. 90 minutes) calls, we argue that VoIP is not yet feasible over (raw) Tor. We identified three reasons explaining why our analysis differs. (1) They do not account for T_{frame} in their analysis. Here we chose $T_{frame} = 40$ ms, our max acceptable delay is then 360 ms. (2) They consider the average delay instead of the 99^{th} percentile one. While we obtain similar average delays as theirs, plotting 99^{th} percentile delay shows that 20% of calls have unacceptable latency for 30-second calls. (3) They consider only 30-second calls while the average call duration is 3 minutes and a significant

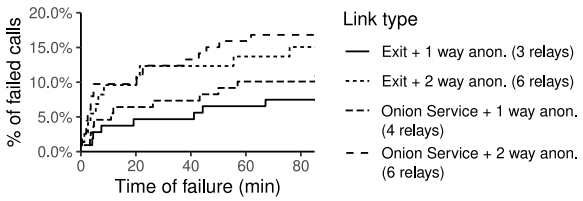

Figure 3.4 – Failed Tor links over time.

share of calls last more than 90 minutes. Measuring delay over a longer timespan showed that delays increase over time.

Comparing the different configurations, we observe that no link offer acceptable delays. We note (Figure 3.3.{B,D}) that the latency benefits from using the *Exit* mode mostly vanish when considering 2-way anonymity. Using one-way anonymity with the *Onion Service* mode (Figure 3.3.C) does not seem to improve tail latency, we presume this is due to the fact that this feature is still experimental.

Moreover, not all link types are equals: using *Exit* links has two drawbacks. First as it requires the last relay of the circuit holds the *Exit* tag. As *Exit* links can send data on the regular Internet, the last relay is particularly sensitive: only 25% of the relays accept to have this position. For the user perspective, it eases de-anonymization attacks and by limiting scalability of the network, harm performances. Moreover, using *Exit* links require to relay the traffic through an ad-hoc public server (eg. Piyush Kumar et al [204] used Mumble and Freeswitch PBX) that must be trusted.

Considering that: (i) no link over Tor enables VoIP, and (ii) *Exit* mode has severe limitations, we choose to focus solely on leveraging *Onion Services* to provide anonymous voice calls in the rest of this paper.

Connection stability. Tor links can not only have latency spikes, but can also break. We evaluate the reliability of each Tor-link types over our longest considered call duration (90 minutes). Figure 3.4 reports the cumulative rate of failed links (i.e., for which packets are no longer transmitted) as a function of time. After 10 minutes, all link types have at least 4% failures. The ratio rises between 7% and 16% after one hour. The failure difference between each link type seems to be correlated with their number of relay: more there are relays, more there are failures. None of the available links satisfy our QoS requirements: we need a solution to overcome link breakage to let the call continues seamlessly.

Predictability of high latencies. The previous experiment shows that the distribution of latency across multiple links is highly skewed. We now evaluate if this skew results

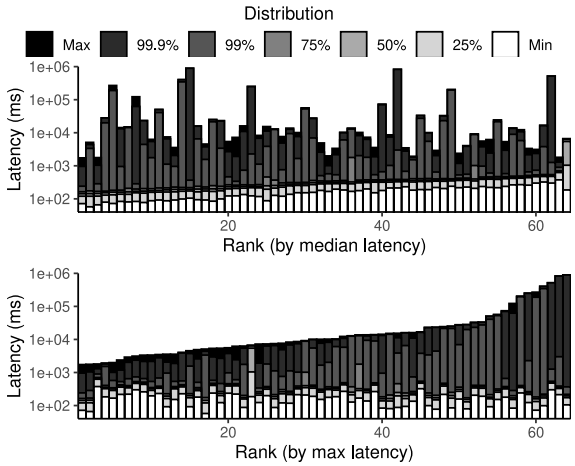

Figure 3.5 – Tor links' latency distribution at 25 pkt/sec ordered by median (top) and max (bottom) latency.

from a large number of poorly performing links with a few, identifiable, good links, or if any link can experience periodic latency bursts. Figure 3.5 presents the one-way-latency distribution for each of the 64 links, ranked by median latency (top) or max latency (bottom). There is no clear relationship between the general performance of a link and the occurrence of latency spikes. The maximal latency does not seem to depend much on the rest of the distribution and can reach very high values in all cases (often 3 times higher than the 75^{th} percentile) [3]. We refer to these high latency periods as *latency spikes* in the rest of this paper.

Discussion. Our experiments confirm the general unpredictability of the performance of Tor links. Due to Tor's exclusive support for TCP [4], latency spikes for a single packet result in high latency for all following packets, delayed to be delivered in order—a phenomenon referred to as *head of line blocking*.

We observe, however, that the number of relays correlates with the probability of networking problems: higher number of relays are associated with higher failure rates or with latency spikes. We note also that most links provide good performance for a fraction of their use time, and failures across links do not seem to be correlated. As a result, we

3. This unpredictable performance is confirmed, in fact, by a blog post by the Tor project [64]. We quote: "*While adding more relays to the network will increase average-case Tor performance, it will not solve Tor's core performance problem, which is actually performance variance.*".

4. TCP maps well to an efficient implementation of onion routing, i.e., allowing to know when to create and dispose of circuits and disallowing packets that are untied to an existing circuit. UDP would also pose security challenges, e.g. enable DDoS attacks. The designers of Tor have clearly dismissed any support of UDP in Tor in the future [205].

make the case for using multiple links, benefiting from periods of good performance, and quickly switching links when experiencing latency spikes.

3.3 Donar: Enabling VoIP over Tor

Donar operates as a proxy between a VoIP application and the Tor client. It does not require modifying either of the two systems. Donar runs without any specific privileges; it only offers a UDP socket to the VoIP application's RTP protocol and opens TCP sockets to the local Tor client. In conformance with our objective to make anonymous VoIP available with *readily-available* systems, we do not require the deployment of an external support service. In particular, Donar does not rely on the SIP signaling protocol but leverages instead Tor onion addresses to establish communications without leaking metadata about communicating parties.

Redundancy by piggybacking. Donar leverages the fact that Tor only transmits data in the form of fixed-sized cells. Setting OPUS to the target bitrate of 32 kbps and using a sending period of 40 ms results in 172-Byte frames. The Tor client pads the remaining space with random data to reach a cell size of 514 Bytes. Donar leverages, instead, this space to re-send the previous frame without changing the necessary bandwidth requirements [5]. Naturally, a redundant frame must be sent on a different link than the first copy, to avoid head-of-line blocking between replicas. While redundant frames are subject to an additional T_{frame} latency (40 ms in the presented configuration), our rationale is that this latency combined with that of the link itself will still be lower than that of a link experiencing a latency spike. We detail next how we effectively enable link diversity.

Link Diversity. Donar leverages multiple Tor links to multiplex traffic in two complementary ways. First, it spreads frame copies onto different links. This prevents packets containing subsequent frames from being subject to the same latency spike thereby arriving too late in a burst at the destination. This also lowers the load on each individual link (resulting, as shown in Section 3.2, in better availability). Second, Donar ensures that the first and the second (redundant) copy of a given frame always travel on different links.

Enabling diversity requires (1) maintaining a set of open links and monitoring their

5. We are not limited to this configuration, and only require that the size of the frames emitted by the codec be less than half the available space minus the Tor headers (8 Bytes) and Donar metadata (38 Bytes in the default configuration), i.e. less than 233 Bytes.

performance; and (2) implementing a scheduling policy for selecting appropriate links for new packets. In the following, we detail these two aspects (§3.3.1 and §3.3.2) and complete the description of DONAR by detailing how calls are established (§3.3.3).

3.3.1 Link monitoring and selection

DONAR opens and monitors a set of Tor links and associates them with scores reflecting their *relative* latency performance. We start by detailing how latency scores are collected at the local client side, and why they must also be collected from the remote client. We motivate our choice to classify links in performance groups, and how we dynamically select links in these groups throughout a call.

Measuring latency. Measuring transmission delays for packets sent over Tor is not straightforward. The RTP protocol uses UDP and does not send acknowledgments. We do not wish to add additional acknowledgment packets over Tor to measure round-trip times, as their padding in 514-Byte cells would result in twice the bandwidth consumption.

Rather than attempting to measure the *absolute* latencies of links, we leverage the use of multiple links to approximate their *relative* latency performance. Measures of performance are continuously collected on both sides of the communication, which we denote as node A and node B in the following. Local *aggregate* measures are then computed over a time window of duration w. We explore the impact of durations ranging from 0.2 to 32 seconds in our evaluation.

We base our measurements on an *out-of-order* metric for VoIP frames. This metric denotes, for an incoming frame f with sequence number i, the number of frames received *before* f with a higher sequence number than i. From the ordered delivery of TCP, these frames are received on different links. For instance, if node A receives frame f with sequence number i from node B on link l after receiving frames with sequence numbers $i+1$, $i+2$ and $i+3$ on other links, we associate an out-of-order metric of 3 to frame f.

The local calculation of the out-of-order metric also applies to *missing* frames. Node A is aware of any *missing* frame f_m with a sequence number $i_m < i_c$ where i_c is the largest sequence number among all the frames received from node B. However, since the decision on which link a packet is sent is made by node B, it is not possible for node A to assign f_m's measurement to a specific link. To solve this problem, we include, in the DONAR headers in each packet, the list of links used for sending the last n frames, where n is the maximum number of links used.

Nodes A and B must share their local aggregate measures to enable fast detection of latency spikes. Node A's local information about a link l approximates, indeed, the one-way latency from B to A, but not from A to B. Our experimental evaluation has shown that one-way latencies are highly consistent in both directions of a link, making node A's local estimation a good approximation also for the latency from A to B. However, this local approximation may be missing if the link has not been used recently by B to send packets to A. We alleviate this problem by embedding, in the DONAR metadata sent with each packet, the local aggregate measures for links that have been measured recently. Node A computes a final array of measures that include, for each link, either (1) the local aggregate measure only, if no remote aggregate was received; (2) the remote aggregate only, if the link was not recently used by B to send data to A; or (3) the average of these two measures if the link was used in both directions.

Link selection. Every w seconds, DONAR sorts links in decreasing order of aggregated scores over the last period, and assigns links to three groups. The $L_{1\text{ST}}$ (first-class) group contains the $n_{1\text{ST}}$ *fastest* links. The $L_{2\text{ND}}$ (second-class) group contains the $n_{2\text{ND}}$ following links. Typically, we use the same number of links in the two groups, i.e., $n_{1\text{ST}} = n_{2\text{ND}}$. Finally, the remaining $n_{\text{INACTIVE}} = n_{\text{LINKS}} - n_{1\text{ST}} - n_{2\text{ND}}$ slowest links are assigned to the L_{INACTIVE} group.

The rationale for using these three groups is as follows. Links in the L_{INACTIVE} group generally experience sub-par performance and remain idle. Links in the $L_{1\text{ST}}$ group have good performance, and are invaluable in allowing fast delivery of VoIP packets. However, the number of good-performing links is limited at a given point in time, and using them systematically bears the risk of overloading them, resulting in lower performance and reliability (§3.2). Links in the $L_{2\text{ND}}$ group are less performant, but remain usable, and can reduce this risk of overload.

Links opening and maintenance. DONAR uses standard operations of the Tor client to open links. It lets the client select relays according to Tor rules. The client allows users to parameterize the number of used guard relays, as well as the length of the links (number of relays).[6] DONAR leverages these parameters to enable different security/performance tradeoffs. We defer the discussion of strategies for setting these values and their security implications, to Section 3.4.

When starting a call, DONAR opens $n_{\text{LINKS}} = n_{1\text{ST}} + n_{2\text{ND}} + n_{\text{INACTIVE}}$ links and assigns

6. The number of guards can be configured as a command line parameter or in a configuration file. The number of relays can be set through Tor client's control port.

them randomly to the three groups. When the Tor client notifies a link failure, DONAR simply requests a new link and assigns it to the L_{INACTIVE} group.

Links in the L_{INACTIVE} group will not be monitored locally. Some of these links may be associated with a remote score, but others will not be monitored on either sides of the call. To enable *all* links to be monitored periodically, we implement a promotion and demotion mechanism between the $L_{2\text{ND}}$ and L_{INACTIVE} groups. When assigning links to groups at the end of a w seconds period, DONAR picks the worst-performing link from the $L_{2\text{ND}}$ group and demotes it to the L_{INACTIVE} group. In return, it promotes to $L_{2\text{ND}}$ the link from the L_{INACTIVE} group that has been unused for the longest time.

3.3.2 Scheduling policies

The DONAR scheduler receives UDP RTP packets containing a single frame from the VoIP application. It first implements redundancy by piggybacking over the pad space, then adds the necessary metadata, and finally creates a TCP packet to be sent onto links from the $L_{1\text{ST}}$ and $L_{2\text{ND}}$ groups.

DONAR's default scheduling policy is named ALTERNATE. It sends each new packet to a *single* link. In doing so, it *alternates* between links from the $L_{1\text{ST}}$ and $L_{2\text{ND}}$ groups. This complies with the requirement to send the first and redundant copies of a frame on different links. DONAR picks the links from each group using a round-robin policy, thereby complying with the requirement of maximizing diversity.

We implement a second policy named DOUBLE-SEND. As the name implies, this policy selects *two* links for sending each new packet. Each frame is received four times: two as a primary copy, and two as a duplicate. This policy doubles the required bandwidth, but has a higher chance to select a fast link for the primary copy of a frame, thereby reducing the risk of delivering the frame with an additional delay of T_{frame}. We note that the resulting bandwidth is the same as for TorFone [145]'s Duplication mode, which systematically sends VoIP packets onto the same two links.

3.3.3 Establishing communication

DONAR leverages Tor's mechanisms to allow callers and callees to establish a connection anonymously. Following our design goal of using only readily-available systems, we do not require the deployment of an existing or novel signaling protocol and, in particular, we do not use a SIP deployment. SIP requires, in fact, the use of trusted proxies and has been

documented as leaking metadata to network observers [143, 144]. Furthermore, with the exception of the audio codec negotiation, SIP functionalities largely overlap mechanisms already offered by Tor [143, 144].

A caller can discover a callee by looking up a specific onion-service identifier using the Tor DHT. This onion service identifier is obtained by other means, e.g. by using an anonymous chat service. The identifier can also be public while still preserving anonymity, as Tor prevents an external observer from determining that a specific client opens a circuit to a specific onion service. For instance, journalists could advertise an anonymous onion service for whistleblowers. We note that client-side authorization, as defined in the Tor rendezvous specification [174], could enable a callee to only allow calls from a whitelist of callers, but we leave the integration of this functionality to future work.

In the current DONAR implementation, the codec and its configuration are hardcoded. Codec and configuration negotiation require, unlike discovery, only communication between the two parties, and could employ a protocol similar to the subset of SIP dedicated to this task. We also leave this implementation to future work.

3.4 Security

DONAR leverages Tor without deploying additional infrastructure or modifying Tor itself. As a result, DONAR inherits the security assumptions and shortcomings of Tor. For instance, like Tor, DONAR does not provide protection from adversaries that can control the *entire* network [199, 176] to perform traffic-correlation attacks [165, 206]. Nevertheless, in terms of guarantees, it is reasonable to wonder whether DONAR worsens the security properties of Tor and to what extent.

In the definition of the so-called predecessor attack, Wright [165] observed that repeatedly creating new circuits causes clients to continuously degrade their security, while increasing the probability that they will eventually choose a malicious relay as the first node of a circuit. Wright [166] proposed to address this problem by using what are now known as guards. Specifically, each Tor client chooses a small number of guards and uses them for all the circuits it ever creates. This suggests that DONAR's impact on security depends mainly on the fact that it can use a larger number of guards than the standard Tor implementation. We evaluate this impact from the perspective of three threats: (1) one endpoint deanonymizing the other, (2) an attacker controlling some relays or AS's identifying DONAR users, and (3) the same attacker deanonymizing both endpoints of a

call and finally breaking anonymity.

Deanonimizing the other endpoint. According to the classification in [32], sender/recipient anonymity refers to the ability to hide one endpoint's identity from the other. As discussed in [166], in a system with c corrupted relay nodes out of n and 1 guard per user, the probability of an endpoint's de-anonymizing the other is $\frac{c}{n}$. If we increase the number of guards to g, this probability becomes $1-(1-\frac{c}{n})^g$, which, for small values of $\frac{c}{n}$, can be approximated from above by its first-order Taylor/Maclaurin expansion $g\frac{c}{n}$. Like most previous work, this analysis focuses on a random distribution of compromised guards. Adversaries can also leverage path selection algorithms to strategically place malicious guards and increase their probability of being selected although countermeasures exist [167].

Identifying Donar users. Identifying a Donar endpoint is equivalent to de-anonymizing any onion service, i.e., identifying which client node is reachable through this service. An adversary controlling a guard relay and knowing the onion address of a callee may observe traffic and employ traffic fingerprinting techniques [168, 171, 172, 169, 170] or use a fake Donar client and perform timing attacks [207] to identify that a specific client is accepting Donar calls. The use of several (g) guards in Donar also increases the probability of this attack to $1-(1-\frac{c}{n})^g$, and thus by a factor of g for small values of $\frac{c}{n}$ like for the de-anonymization of one endpoint. This attack can however be mitigated by using the client-authorization feature offered by V3 Onion Services [174]. Finally, while several authors have shown that an adversary could locate onion service endpoints even when they were not publicly advertised [173, 171, 172, 168], they have also proposed solutions to the Tor community.

De-anonymizing an ongoing call. To de-anonymize an ongoing call, an attacker needs to control guard nodes at both endpoints and employ traffic correlation techniques [206]. As a result, like for the first two threats, the choice of the number of guards used by Donar identifies a tradeoff between the likelihood of this attack and the performance of a call. In particular, since the attacker needs to control at least one guard on each side of the call, the associated probability grows from $(\frac{c}{n})^2$ with one guard to $(1-(1-\frac{c}{n})^g)^2$ with g guards. This implies that it grows even more slowly for small values of $\frac{c}{n}$ than the two previous probabilities.

Finally, we also observe that passive traffic correlation attacks turn out to be more difficult to perform when multiple calls are ongoing as Donar's traffic patterns do not vary between different calls. In this case, a passive attack must observe the start and/or the end of a call to be effective.

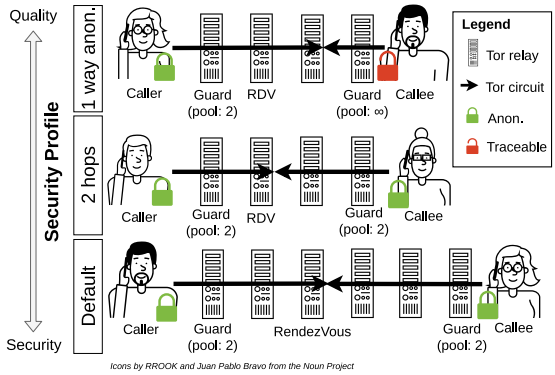

Figure 3.6 – Security configurations.

Donar security configurations. As discussed above, increasing the number of guards improves performance but it also increases the attack surface. For this reason, Donar implements three security configurations that strike different tradeoffs between privacy and performance, as illustrated in Figure 3.6. We emphasize that each configuration sets up the Tor client via its legacy API, and hence does not require modifying the legacy Tor client. In all configurations, Donar uses 12 links, but link settings are different in each configuration. The *Default* configuration provides a security strengh similar to the legacy Tor client with default Tor link settings, i.e. each link has 6 relays, and each client employs only 2 guards.[7] The *2 hops* configuration sets up the Tor client so that each created link has two fewer Tor relays compared to Tor's default link settings. Finally, the *1 way anonymity* configuration totally remove anonymity of the callee, that will use a *single* Tor relay (without the guard pool constraint) between the callee and the rendezvous point.

Security Discussion. Each of the threats we identified above relies on the control of at least one guard relay per affected endpoint. As discussed above, Donar does not modify the guard configuration when providing anonymity for a user. Moreover, the use of guards decorellates the number of links and the de-anonimization probability: using 12 links at once does not expose more a user that using only one. Additionally, compared to the *Default* configuration, the *2 hops* one reduces the number of relays in links by two. Decreasing the number of relays in links has been long debated in the Tor community. The main rationale for using 3-relay circuits (and thus 6-relay links) is that it makes it more difficult for an adversary that controls the last relay to identify the entry guard. On

7. Even though Tor's documentation discusses using only one guard, the default client uses two.

the other hand, an adversary can overcome this protection with relatively low investment in additional relays, and 3-relay circuits are more vulnerable to attacks based on denial of service [208]. These observations motivate our choice of 2-relay circuits with better latency in our *2 hops* configuration. Finally, the *1-way anonymity* configuration does not provide anonymity to the callee but does not incure any change to the caller. Moreover, this mode is a standard feature of the Tor daemon that is used in production (eg. by Facebook [**noauthor_facebooks_nodate**]).

Finally, we emphasize that DONAR users may also explore entirely different security configurations, by changing the number of Tor guards and/or relays for links, according to their own expected tradeoffs between performance and security.

3.5 Evaluation

We implemented DONAR and will release our code as open source with the unblinded version of this paper. The DONAR proxy interfaces a VoIP application with the Tor client. [8] We use two applications: (1) a configurable RTP emulator allowing a fine-grained control on the frames sent between parties, and running multiple occurrences of an experiment to study statistical variations; and (2) the actual `gstreamer` VoIP application using the OPUS codec. We deploy two isolated instances of either application on the same machine to accurately measure one-way delays for packets sent over Tor.

Tor's performance varies over time, with failures, disconnections, and latency spikes as identified in Section 3.2. Unless mentioned otherwise, we run each experiment a total of 64 times and present the distribution of results. We run the same configuration over a long time span, typically 5 hours, and also compare different configurations running in parallel.

3.5.1 Performance & SOTA comparison

We start with the evaluation of the global performance of DONAR and its ability to meet the requirements summarized in Table 3.1. We use an audio stream of 32 kbps with a rate of 25 frames per second. We configure DONAR as follows: The window duration is $w = 2s$ and we open a total of $n_{\text{LINKS}} = 12$ links including $n_{1\text{ST}} = 3$ links, $n_{2\text{ND}} = 3$ links,

8. The Tor software is evolving quickly, especially considering v3 onions. To benefit from latest bug fixes, we compiled Tor from branch `maint-0.4.4` (commit `09a1a34ad1`) and patch #256.

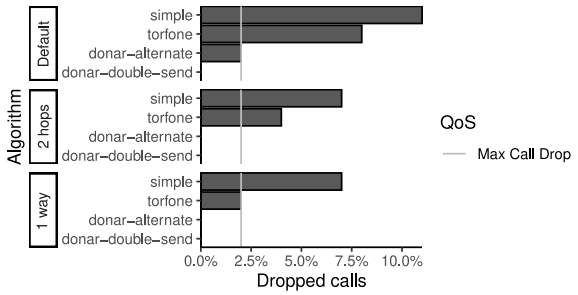

Figure 3.7 – Dropped calls after 90 minutes for SIMPLE, TORFONE, and DONAR setups.

and $n_{\text{INACTIVE}} = 6$ links. We present a comprehensive analysis of the influence of these parameters in Section 3.5.2.

We consider the six possible variants of DONAR using either of the two scheduling policies ALTERNATE and DOUBLE-SEND combined with one of the three security configurations (*Default*, *2 hops*, or *1-way anonymity*). In addition, we implement two approaches representing the state of the art. SIMPLE is the direct use of a single Tor link to transfer VoIP data. It represents our reference in terms of bandwidth usage for the ALTERNATE policy. TORFONE implements the duplication strategy used in TorFone [145]: It sends each new packet on two links, representing a reference for bandwidth usage for the DOUBLE-SEND policy.

No call interruption. We start by studying the percentage of dropped calls for all configurations. We run 96 instances of a 90-minute call for each combination and count the percentage of dropped calls. For SIMPLE, a broken Tor link invariably results in a dropped call. The DONAR variants and TORFONE, instead, re-establish broken links, and thus consider their calls dropped whenever they miss 25 consecutive frames. Figure 3.7 presents the results. All DONAR variants perform better than the previous approaches, and meet the goal of less than 2% of dropped calls. We only record, in fact, dropped calls for the most conservative of our setups, i.e., combining the ALTERNATE policy with the *default* configuration. TORFONE only meets the goal in the *1-way anonymity* configuration.

Interactive conversations & good voice quality. These objectives require a sufficient bitrate—met by using a 32 kbps bitrate in our experiments—and receiving at least 99% of VoIP frames within the maximum acceptable latency. The OPUS codec can, indeed, mask the loss of 1% of the frames with no perceptible quality degradation.

We present the distributions of frame delivery latencies in Figure 3.8. Our mouth-to-ear latency objective is 150 ms, and our limit is 400 ms. As T_{frame}=40 ms, we wish network

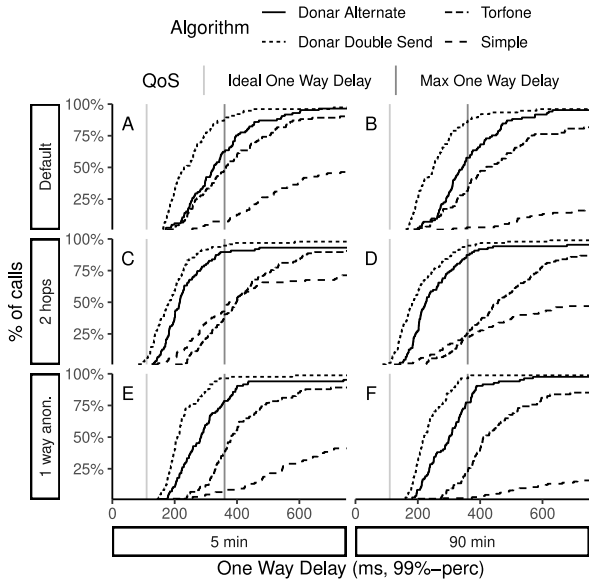

Figure 3.8 – Latency comparison between SIMPLE, TORFONE, DONAR ALTERNATE and DONAR DOUBLE-SEND.

delays for delivering frames to be of 110 to 360 ms. We use two vertical lines to denote these boundaries.

For all security policies and call durations, the DONAR DOUBLE-SEND algorithm provides at least 87% (*Default*, 90 minutes) of successful calls. Considering only our optimized security policies, the ratio of successful calls is even higher at 95%. These results must be compared to TORFONE, as they both send the same amount of data on the wire. TORFONE enables as low as 23% (*1 way anon.*, 90 minutes) and at most 47% (*Default*, 5 minutes) of successful calls. Comparing on DONAR DOUBLE-SEND's worst performance (*Default*, 90 minutes configuration), there is a 55 points difference with TORFONE in favor of DONAR.

Conversely, we observe that DONAR ALTERNATE does not fit all configurations: for its *Default* security policy, it enables only 62% (resp. 57%) of successful calls for 5 minutes (resp. 90 minutes). Results are better with the *1 way anon.*: 78% (resp. 77%) for 5-minute (resp. 90-minute) calls. However, only the *2 hops* configuration seems to offer acceptable quality, enabling at least 87% of successful calls. Compared to the SIMPLE mode that the sends the same amount of data, it is gain of 55 points for DONAR worst performance. With the *2 hops* configuration, it is a 43 points (resp. 65 points) for 5-minute (resp. 90-minute) calls improvement on SIMPLE.

To conclude, DONAR DOUBLE-SEND is able to offer a high ratio of successful calls in any situation (87%+ compared to 23%+ for TORFONE); it is a versatile solution at

the cost of adding redundancy on the wire. In comparison, DONAR ALTERNATE has no overhead but is way more sensitive to the configuration: it only works well with the *2 hops* security policy (87%+ compared to 46%+ for SIMPLE). With a difference of at most 4% between the 5-minute and 90-minute measurements, DONAR adds a new interesting property: latency stability over time. We argue that our two sending policies represent a significative improvement in term of delay compared to the state of the art.

Using the `gstreamer` VoIP client. We performed experiments with the replay of an audio file using the `gstreamer` VoIP application. We collect statistics about its jitter buffer. `gstreamer` only allows a static-size jitter buffer. We configure this buffer based on our previous experiments, so as to absorb latencies between the minimum observed latency and the 99^{th}-perc. latency, and count the number of calls that systematically meet latency requirements out of the 64 experiments done for each configuration. Our results confirm that DONAR DOUBLE-SEND is able to meet the 360 ms latency threshold for most experiments in all configurations, while the ALTERNATE policy works best under the *2 hops* configuration. We also confirmed empirically the results obtained under the *2 hops* configuration and the two scheduling policies by performing actual calls between two laptops: we could not detect any degradation in sound quality throughout any of the calls.

3.5.2 Microbenchmarks

In the following, we present an analysis of the influence of each of DONAR's parameters, and of the complementarity of its mechanisms. We focus on the six possible DONAR variants and, to factor out the impact of security configurations, we also consider a version of DONAR using 4 relays per link and an unlimited number of guards.

Protocol parameters. DONAR has 3 main parameters: w, n_{LINKS}, $n_{1\text{ST}}$ (we use $n_{1\text{ST}} = n_{2\text{ND}}$). In the experiments reported in the previous section, we employed the default values of $w = 2s$, $n_{\text{LINKS}} = 12$ and $n_{1\text{ST}} = n_{2\text{ND}} = 3$. We detail in the following how we selected this default configuration.

We present in Figure 3.9 an analysis of the influence of each parameter on the distribution of frame delivery latencies. Parameter w determines how far in the past we consider out-of-order metrics when computing links scores. It also determines how many times we need to probe a link before deciding to stop using it. A lower value of w enables fast reaction at the risk of too many links switching and unreliable scores, while a larger value

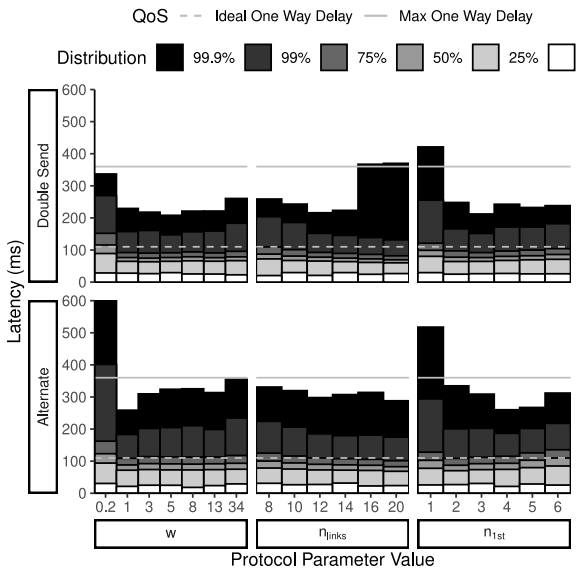

Figure 3.9 – Impact of protocol parameters (w, n_{LINKS} and $n_{1\text{ST}} = n_{2\text{ND}}$) on frame delivery latencies.

promotes links that are stable over time. We can observe on the left side of Figure 3.9 that the best value of w for the DOUBLE-SEND policy is 5s, while the best for the ALTERNATE appears to be 2s. Additional benchmarks on the $[1, 8]$ range with a smaller step lead us to select the latter value as the default.

The n_{LINKS} parameter controls the total number of open links and, therefore, both the level of achievable diversity and the load of route maintenance on the Tor network. We evaluate n_{LINKS} values from 8 to 20. The ALTERNATE policy performs best with 20 links, while the DOUBLE-SEND policy performs best with 12 links. To limit the load on Tor, we select this latter value as the default.

Finally, parameter $n_{1\text{ST}} = n_{2\text{ND}}$ directly controls the number of links that are actively used to send packets. On the one hand, for a given value of n_{LINKS}, a small value of $n_{1\text{ST}}$ increases the likelihood of selecting only good-performing links. On the other hand, a large value increases diversity and the frame rate on each link, resulting in higher stability as we have shown in Section 3.2. Using $n_{1\text{ST}} = 1$ yields high latencies with either variant, while $n_{1\text{ST}} = 3$ or $n_{1\text{ST}} = 4$ offer a good compromise. We choose $n_{1\text{ST}} = 3$ as our default value.

Impact of the size of the guard pool. We considered using different sizes for the guard pool, for the different security configurations detailed in Section 3.4. We further explore the impact of this parameter on DONAR performance. Our results, shown in Figure 3.10,

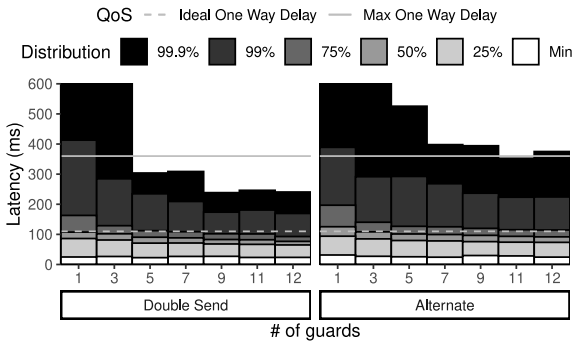

Figure 3.10 – Impact of Tor guards number on latencies.

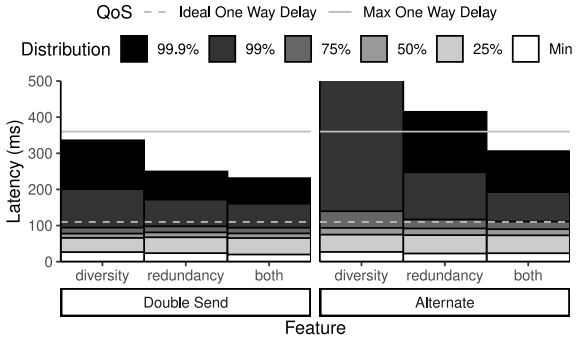

Figure 3.11 – Diversity & redundancy complementarity.

confirm that, in order to achieve the best latency, it is preferable to have as many guards as the number of links, in our case $n_{\text{LINKS}} = 12$. This number is, however, the result of a compromise with attack surface. In our performance evaluation, we chose to stay conservative and do not modify the guard number but we demonstrate here this choice has a performance cost.

Complementarity of diversity and redundancy. We analyze to which extent the two enabling mechanisms of DONAR, diversity and redundancy, contribute to its performance. We present latency when using only link selection (diversity), using only redundancy by piggybacking, and using both, in Figure 3.11. Activating both features is clearly beneficial for both scheduling policies, but, unsurprisingly, the impact of redundancy by piggybacking on high percentiles of the distribution is larger for the ALTERNATE strategy than for the DOUBLE-SEND strategy, as the latter enables redundancy by sending packets twice.

We further wish to understand how diversity and redundancy interact when used simultaneously, by analyzing, for each frame, which group of links delivers it for the first time, and whether this first delivery concerns a primary or a duplicate copy. The first

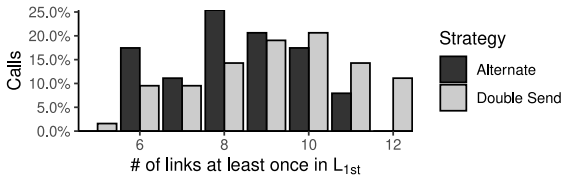

Figure 3.12 – How many links were L_{1ST} at least once?

delivery of a frame, indeed, results from a *race* between two send operations (with the ALTERNATE policy) and four send operations (with DOUBLE-SEND).

When using the ALTERNATE policy, 94% of the primary frames copies sent on a link of the L_{1ST} group arrive first, and only 6% are received as a duplicate copy via an L_{2ND} link, despite being sent 40 ms later. When the primary frame copy is sent over an L_{2ND} link, however, only 48% arrive before the duplicate copy sent over an L_{1ST} link, while as many as 52% arrive as a duplicate copy, again despite being sent 40 ms later. When using the DOUBLE-SEND policy, 73% of the frames are received first as a primary copy on the L_{1ST} link, 14% are received as a primary copy on an L_{1ST} link, and only 13% are received as a duplicate copy. Using L_{2ND} links remains useful. It provides more diversity through the use of more links, while still leveraging the reliability of the best links. Moreover it decreases the load on each individual link, reducing the risk of performance degradation on each of them.

Link monitoring effectiveness. We finally evaluate link monitoring, and assess whether link classification and selection reflect the behaviors discussed in Section 3.2.

We start by observing the distribution, over 64 calls, of the number of links that were classified as L_{1ST} *at least* once through the duration of a 90-minute call. This distribution is given by Figure 3.12. Note that we do not consider the first 40 seconds of each call, as DONAR has to bootstrap the process with random scores, and poorly-performing links could be assigned to the L_{1ST} group during this bootstrap. Between 6 and 12 links per call have been considered at least once in the L_{1ST} group in every call, with a majority of 8 to 10 links selected. This confirms our analysis that there is no single link that is consistently performing well in Tor, and that link performance varies significantly over time: Links that are poorly performing at a given time may be the best ones a few minutes later.

We study, in finer detail, the stability of links over time, focusing on a single call using the ALTERNATE policy with the Default configuration. We represent the latency of the first delivery of each frame in the first plot of Figure 3.13. This is the latency that

is observed by the VoIP application. Latency remains low throughout the call. In the second plot, we decompose the latency of frames received on the L_{1ST} and L_{2ND} groups, including the first and second receptions. We can clearly see that the latency of the links in the L_{1ST} group is generally lower, and that outlier values are compensated by lower latency on a link in the L_{2ND} group. The third plot represents the assignment of the 12 links to link groups over time. We note that there was no link failure (and therefore no link replacement) in this experiment. Link 0 is, for instance, classified in L_{1ST} for a large part of the call, but suffers a latency spike around frame 6,500 and is rapidly classified in the $L_{INACTIVE}$ group. Link 2, initially in $L_{INACTIVE}$, is promoted 3 times with no effect to the L_{2ND} group, before being selected as L_{1ST} after its fourth promotion. Links 1, 5, 7 and 8 have highly heterogeneous behaviors, while links 3, 4, 6, 11 and 12 have consistently bad behaviors, and only appear in the L_{2ND} group upon their promotion before being quickly deactivated. While these links could be proactively replaced by opening new links, we do not deem it necessary and choose not to impose further link setup load on the Tor network.

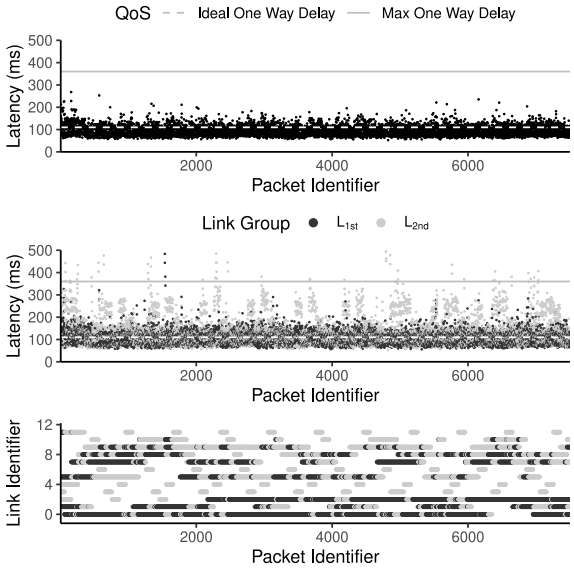

Figure 3.13 – Stability over time.

3.6 Conclusion

We presented Donar, a solution for high-quality VoIP calls. Donar enables readily-available anonymous calls using the challenging but existing Tor network. Donar circumvents Tor's inability to support the networking requirements of VoIP by sending audio frames over a diversity of links and using redundancy without incurring any additional bandwidth costs. It offers different tradeoffs between performance and security, and successfully enables high-quality VoIP calls, e.g., with latency below 360ms during an entire 90-minute call.

This work passed NSDI 2021 conference's first round review but we are still waiting for the final decision.

After studying low-latency enhancements to anonymity networks, we focus our work on enabling high-throughput applications.

Chapter 4

High-Throughput Communication Over a New Edge-First Onion Protocol

4.1 Introduction

In this chapter, we present eTor, an enhancement of Tor, to leverage residential relays that are already present in the network. At the core of the eTor proposal lies the novel concept of an *Intertwined Onion Circuit* (`IOC`). Each hop of an `IOC` aggregates multiple edge relays, instead of just one in a traditional onion-routing protocol. When receiving a packet, an edge relay can leverage operational conditions (congestion, availability, latency) to select the next best relay among those contained in the `IOC`. Multiple relays may be used simultaneously to support traffic *multipathing* and increase the `IOC` capacity. This flexibility in dynamically selecting relays allows `IOC` to offer continuous and high-capacity communication over relays subject to churn. The potentially large pool of edge relays have the potential to considerably extend Tor's pool of usable relays, and hence its peak bandwidth capacity.

eTor preserves Tor's original anonymity properties: the negative impact caused by the presence of multiple relays per hop is compensated by the greater number of relays that become usable in the network. In particular, we show that for a population of relays experiencing high churn, it is better for the sake of anonymity to include multiple volatile relays at each hop rather than ignoring them.

In summary, our contributions are as follows:

— We introduce a novel onion routing mechanism that extends Tor, based on the new concept of an *Intertwined Onion Circuit* (`IOC`) that: (i) leverages multipathed onion circuits formed of edge relays, to provide continuous availability and better

capacity, and (ii) leverages availability prediction to enable the selection of an optimal number of such edge relays per circuit hop while preserving anonymity.
— We present how IOC can be integrated into the existing Tor to form eTor, preserving the security guarantees of Tor.
— Finally, we demonstrate that using IOC paves the way to the use of anonymous bandwidth-intensive communication without disrupting Tor. We evaluate eTor both in simulation and in a real deployment. It results that with IOC it becomes possible to involve massively edge relays in eTor while masking the negative impacts of churn.

The remainder of this chapter is organized as follows: we first define our problem in section 4.2, then we introduce our approach in section 4.3 and analyze its security in section 4.4. We present the experimental evaluation of eTor in section 4.5. We conclude this chapter in section 4.6.

4.2 Problem Definition and Goals

Tor exploits a network of relays ideally hosted on dedicated servers with broadband network access. Hosting and operating a new relay requires, indeed, a significant commitment: the Tor community recommends the provision of a server with a symmetric 16 Mb/s upload and download network capacity and with very high availability figures (ideally, the server must be available 24/7). At the same time, the anonymity guarantees that Tor provides depend on network diversity: relays must be spread over as many different infrastructure providers as possible, thereby spanning multiple Autonomous Domains (AS), to be resilient to attacks and outages. There is, as a result, an inherent tension between the incentive to use servers in public (or private) clouds and the need to avoid renting large numbers of servers from the same few cloud providers.

The number of relays exploited by Tor ($\sim$ 6500 active relays) remains small in comparison with the number of active users. Capacity, availability and diversity requirements unfortunately impair the ease by which new relays can be added to the Tor network. This is particularly problematic for resource-intensive application, such as the anonymous exchange of large files. The amount of bandwidth available in the network does not scale with the number of users, and the multi-hop nature of Tor traffic also imposes a high overall resource consumption. Exchanging a file between two users using OnionShare [147] requires, for instance, the use of a circuit formed of 6 Tor relays.

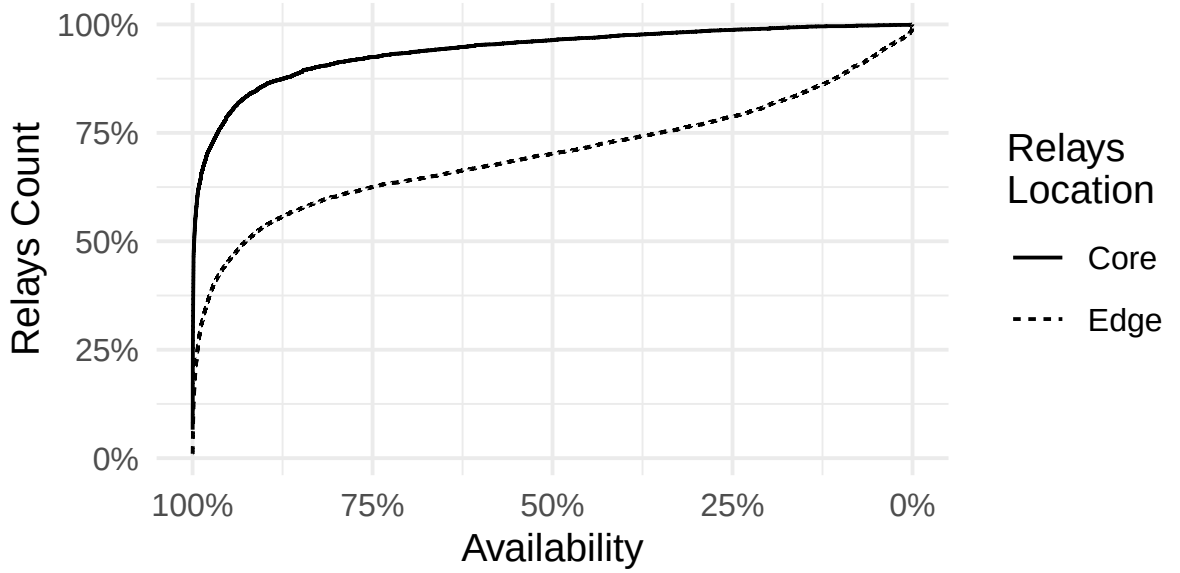

Figure 4.1 – Empirical Cumulative Distribution Function of Relays Availability registered in the Tor relay directory. Data were collected between 2019-05-01 and 2020-05-31.

Fortunately, we identify that untapped resources exist that can be leveraged to extend the anonymity network capacity and support bandwidth-intensive operations. A common characteristics of these resources is that they are located at the *edge* of the anonymity network. In the following, we detail two categories of edge relays using such edge resources: those existing already, but not used to their full potential in the Tor network, and those that could leverage network and computational resources provided by volunteer Tor users themselves.

Existing Tor Edge Relays. Firstly, we identify from an analysis of Tor's history, that a fair number of current relays with low availability figures are in fact deployed *outside* of data centers. We collected from Tor metrics all Tor hourly consensus (list of available relays) over 13 months, from May 1st, 2019 to May 31st, 2020. The dataset contains 32,695 unique relays. Some of these relays are only available for a very short amount of time, possibly deployed for testing purposes only. We only consider, therefore, relays for which the total period of availability over these 13 months has been of at least one week, cumulated ($24\times7=168$ reports). The resulting pruned data set contains 19,585 unique relays. The median number of available relays for each consensus is 6,432 relays. We are interested in studying the correlation between relays availability and the location of the corresponding servers. Location information is not directly available in the consensus,

but we are able to determine using IPHub [209] whether the relays' IP address classifies as residential or lie within a data center. We consider the former as *edge relays*, and the latter are *core relays*. Figure 4.1 presents the Empirical Cumulative Distribution Function (ECDF) of relays availability over the complete 13-month period, i.e., the proportion of consensus instances featuring the corresponding IP address. We clearly observe different trends between edge and core relays. 63.5% of core relays are 99% available over the period, but this number drops to 29% for edge relays. The median availability is about 95% for core relays but only 72% for edge relays. Overall, we can observe a clear correlation between the availability of relays and their core/edge location.

The difference in availability, but also in available capacity, has an impact on the *use* of edge relays. Relays with low availability are generally excluded by clients when forming circuits, in order to enable an initial measurement phase of their capacity. Significant work has been conducted in the last decades to refine relay selection algorithms based on criteria such as availability, latency, bandwidth, congestion, security, and/or location [210, 211, 105, 212, 213]. While improving the quality of Tor circuits for end users, these criteria also reduce the odds that edge relays be selected by Tor clients when creating these circuits, even past the initial measurement phase. This is reflected by the prevalence of relays labeled as guards (i.e., possible entry points for Tor circuits) in the two sets: while 2,272 out of the 3,910 core relays (58%) are labeled as guards, only 736 out of the 2,522 edge relays (29%) are. Existing edge Tor relays are, in conclusion, generally underused compared to their core counterpart.

Future Domestic Edge Relays. Secondly, many Tor users have domestic broadband network access and already host edge devices with unused capacity. This includes, for instance, set-top-boxes for Internet access but also mini-servers (e.g., NUCs) used as IoT hubs or for home automation. For example, a set-top-box may be available for use as a relay during the night or when the primary users of the home network are away. These devices have even lower availability guarantees than existing Tor edge relays: they may be, for instance, only available to act as a relay when not used for their primary purpose. Such domestic edge resources have, however, a significant advantage over dedicated edge resources, which is their potential number. Even if a modest fraction of users choose to dedicate some of their home networking and computational capabilities to the anonymity network, the number of potential relays still grows proportionally to the total number of users.

4.2.1 Functional and performance goals

Our goal is to enable the use of edge relays in Tor, to support additional capacity for the anonymity network. While existing traffic, in its majority Web browsing [**mani2018understanding**], can continue to use the network of core Tor relays, we envision that the set of existing and future edge relays be used to support novel usages, and in particular high-volume anonymous data transfers.

The objective of our work is to extend Tor circuit creation and operation protocols with a mechanism adapted to the nature and characteristics of edge relays. The nature of these relays introduces, indeed, unique challenges if we wish to use them effectively and efficiently. We detail these challenges in the following paragraphs.

Taming (un)Availability. In Tor, regular onion circuits are down as soon as one relay is down, requiring clients to recreate entire new circuits, and breaking the client-side TCP connection. While this is acceptable for Web browsing with short-lived connection and using highly-available core relays, this "weak link" property becomes a significant problem considering, on the one hand, the generally low availability figures of edge relays (churn) and, on the other hand, the longer connections required for file transfers. The odds, for a regular Tor onion circuit over edge relays to break during the said transfers are high. As we do not wish to implement an application-level retry mechanisms (*à la* FTP) but rather to support existing applications using regular TCP channels (e.g., `scp`), we favor masking unavailability events in the constructed circuits from the application, and build highly-available end-to-end circuits over otherwise transient resources.

Offering High Capacity. An expected characteristic of edge relays compared to core relays is their lower bandwidth and network capacity. Oftentimes, domestic edge relays will even be connected through asymmetric network connections offering limited upload capacity. The anonymous transfer of large files should ideally take place with a good bandwidth, ideally reaching the upload capacity of the sender. As for availability, we do not wish to requires changes to the client application, which can open and operate a single TCP channel transparently as for regular, non-anonymous communication.

Staying Compatible with Existing Tor. Finally, our solutions must not introduce elements others than the novel Tor circuit creation protocol and the associated logic at edge relays and directories, and must not prevent the conjunct use of regular Tor circuits. The discovery and selection of edge relays to form circuits shall, in particular, only use Tor existing consensus mechanism over available relays. This selection must also abide

with Tor anonymity and security objectives, as we detail next.

4.2.2 Anonymity goals

The fulfillment of our functional and performance goals should not come at the cost of lower anonymity compared to regular Tor. We detail in the following the anonymity properties that eTor, similarly to the current Tor network, must guarantee.

Adversary model. We consider the same adversary model as Tor. The attacker's goal is to de-anonymize file exchanges endpoints. Its action may target a specific source or destination, e.g., to discover which whistleblowers are sending files to a specific journalist, or be general and attempt to map all or a fraction of the social graph of interaction between users.

We consider that the attacker can control a number of resources (edge relays) and participate to service provisioning, with the goal of breaking regular users' anonymity. The attacker is malicious and can arbitrarily deviate from the protocol, by dropping, replaying, or forging messages. In particular, the attacker may take the role of a corrupt insider to the protocol and attempt to initiate exchanges as if it was a regular user, with the goal of de-anonymizing the communicating partner.

Anonymity properties. We aim to ensure three anonymity properties derived from the AnoA framework [32]: *Sender-Receiver Anonymity*, *Sender Unlinkability*, and *Relationship Anonymity*. In the following discussion of these properties, S denotes a sender, R a receiver, $\{S \rightarrow R\}$ a communication between S and R and $\mathcal{A}$ an attacker.

Sender-Receiver Anonymity. If one of the participant is compromised or under surveillance, we want to protect the identity of the other participant. In a practical case, it could help journalists under surveillance to protect their sources. Formally, given that $\mathcal{A}$ knows the receiver R, $\mathcal{A}$ must not be able to determine S. In particular, $\mathcal{A}$ must not be able to distinguish between two communications $\{S_1 \rightarrow R\}$ and $\{S_2 \rightarrow R\}$. Conversely, given that $\mathcal{A}$ knows the sender S, $\mathcal{A}$ must not be able to determine the identity of R. This means that $\mathcal{A}$ must not be able to distinguish between two communications $\{S \rightarrow R_1\}$ and $\{S \rightarrow R_2\}$.

Sender Unlinkability. It must be impossible to determine if two messages come from the same sender or not. Hence, an attacker cannot infer communication patterns of participants that could reveal information about their behavior. Formally, $\mathcal{A}$ must not be able to distinguish between two communications $\{S_1 \rightarrow\}$ and $\{S_2 \rightarrow\}$.

Relationship Anonymity. It must be impossible to build a social graph by observing communications, hence communications must be indistinguishable. Such a social graph forms precious information and de-anonymization attacks could be possible by crossing some datasets. er ▶*missing reference*◀ Formally, $\mathcal{A}$ must not be able to distinguish between $\{S_1 \to R_1, S_2 \to R_2\}$ and $\{S_1 \to R_2, S_2 \to R_1\}$.

4.3 Approach

We present eTor, our proposed evolution of Tor to enable the use of edge relays for bandwidth-intensive anonymous communication such as file exchange.

4.3.1 Overview

The eTor infrastructure leverages the key concepts of Tor: (i) a relay directory, (ii) service directories, (iii) relays, (iv) users, and (v) onion circuits. Where appropriate, eTor reuses *as is* the building blocks of Tor.

Relay directories. eTor uses as is the Tor's Directory Authorities (DA) allowing relay directories to collect the complete set of relays existing in the system. Periodically, the list of existing relays is updated among relay directories following the Tor consensus protocol. Fully replicating the maintenance of a global view of the network may be foreseen as a possible scalability bottleneck, as the number of relays (core and edge) grows. The problem of scaling up relay discovery without the need to maintain a global state view of the network has been addressed by past work [91, 88] that eTor can leverage if necessary–our current design relies on Tor relay directories with no modifications.

Service directories. eTor reuses Tor onion services to ensure sender-receiver anonymity. It provides anonymity for both interacting users without revealing their network location.

Relays. As mentioned in the previous section, we distinguish between core relays, hosted in data centers (public clouds) and offering good network connectivity, and edge relays, which are either dedicated servers running outside of such data centers (e.g., in a private office deployment) or domestic edge relays operated by individuals using the spare capacity of their home servers and set-top-boxes. We consider that relays are aware of their nature as core or edge relays, and that this information is maintained as well by the relay directories. While core relays do not need to be modified, we assume that edge relays are

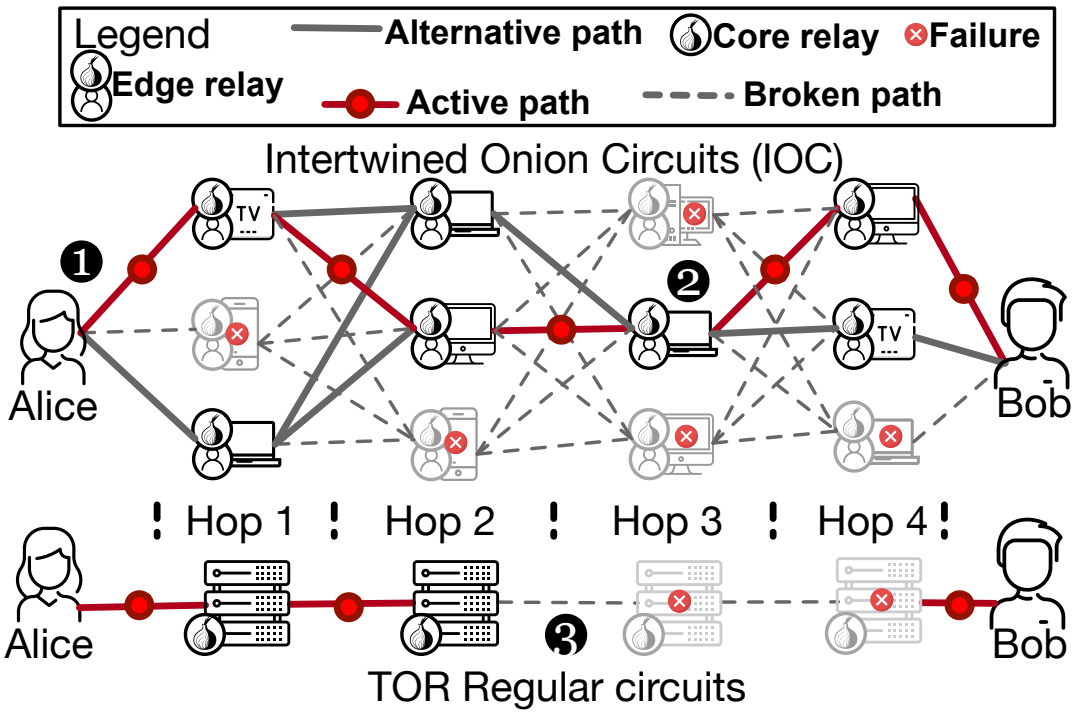

Figure 4.2 – eTor introduces Intertwined Onion Circuits (IOC). IOC are robusts as multiple relays are packed per hop and, in the long run, connections can be restored by relays themselves. IOC are also efficient, as each relay can choose the less congested relay of the next hop to maximize network usage.

aware of eTor specific circuits creation and operation protocols in addition to the Tor regular protocols.

Users. As in Tor, users are ordinary devices used by users to access the network. They are not subject to specific availability, performance, or capacity requirements. Users get descriptors about available relays in the network from the relay directories. They further register or query service descriptors to/from the service directory, preserving user anonymity by performing such query using an onion circuit. Finally, thanks to the service descriptor, an anonymous tunnel can be opened between two participants. Such a tunnel is built by forwarding data amongst multiple relays to prevent an adversary from identifying both parties of the communication.

Intertwined Onion circuits. eTor primarily differs from Tor in the way onion circuits are created and managed, offering two different mechanisms for forming and operating circuits. The first mechanism is the circuit creation and operation of Tor, and allows

Circ ID (I^i)	**Key** (K^i)	**Next Hop** (R^{i+1})	**Previous Hop** (R^{i-1})
43	D78xXGC...	186.192.59.213 12.159.139.74 126.68.197.123	7.60.204.144 228.97.118.107 92.66.162.32
28	2FWqnuc...	67.196.108.223 18.205.24.43 50.54.163.58	40.128.212.96 222.62.226.42 99.25.137.235
...	...	...	...

Table 4.1 – Example of a relay's circuit table (parts in green are eTor's additions)

establishing circuits formed of relays selected from either set of relays–though it is in the interest of clients to select such relays amongst core relays. The second mechanism is specific to eTor, and only uses edge relays. It allows establishing a new form of onion-routing circuit that can overcome the volatility and resource constraints of these edge relays. To answer the availability and capacity challenges highlighted in the previous section, eTor exploit redundancy by packing multiple edge relays to each hop in an onion circuit, forming an Intertwined Onion Circuit (`IOC`) as depicted in Figure 4.2 ①.

An `IOC` offers multiple paths to send data: if one of these paths sees one of its constituents (relays) fail or get temporarily unreachable, then the overall end-to-end circuit remains open as long as there exists at least one valid path (Figure 4.2 ②). This contrasts with regular Tor circuits that use a single-path between the two communicating parties: when one of the relay fails or is unavailable past the detection timeout, a new circuit has to be re-created from scratch and the TCP connection to/from the client is closed (Figure 4.2 ③). In addition to their contribution to robustness, `IOC` enables high capacity circuits by exploiting multipathing: data can be simultaneously sent across different circuits between the source and destination. At each hop, relays are able to select the next hop relay based on the current congestion, further allowing to adapt to unstable edge relay network conditions. We detail in the following the operation and lifecycle management of `IOC`.

4.3.2 Relaying Data In An `IOC`

Onion routing consists of recursively encoding a data chunk first using the receiver's key, then using the last-hop key, and so on until the first-hop key. The resulting data

is referred to as an *onion cell*. Onion routing can be performed in a stateless manner by using relays' public keys to recursively encrypt the payload. The address of the next relay must be embedded in each layer. Public key cryptography is, however, slow and the routing information requires additional bandwidth. To overcome these limitations, Tor uses switched virtual circuits (i.e. onion circuits), where each relay has to maintain a routing table. On receiving a packet at hop i, the relay extracts its circuit identifier I^i. As depicted in Table 4.1, the relay can retrieve the associated secret K^i and its adjacent relays R^{i+1} and R^{i-1} from the circuit table. This information was filled in the table upon the opening of the circuit. In eTor, the use of `IOC` introduces multiple relays per hop, modifying routing information that must be stored by a relay to $[R_1^{i+1} \cdots R_n^{i+1}]$ and $[R_1^{i-1} \cdots R_n^{i-1}]$ where n is the (configurable) number of relays per hop. Additional relays for each hop are colored in green in the table.

eTor, like Tor, uses *control cells* to manage (i.e create, update, destroy, ping) `IOCs`, and *relay cells* to carry end-to-end data. While *control cells* are addressed to only one relay, in eTor, *relay cells* are addressed to *any* relay of a hop. When handling a cell, a relay must now choose the next candidate to which it will forward that cell. To make this choice, the relay reviews all sending sockets associated with the `IOC`'s next hop. The rationale for the selection of the candidate is that it must be available and not congested. To this end, eTor leverages feedback from TCP sockets: availability is estimated from the candidate socket not being broken *via* heartbeats and timeout, while congestion is estimated by observing sending queues [1]. The internal sending queue of a candidate grows if we schedule packets faster on this queue than it is able to deliver them. If all candidate sending queue length exceeds a defined threshold, eTor stop reading incoming packets for the considered `IOC`, waiting for one of the candidate sending queue size to decrease. Such action will increase the prior relay sending queue, which will in their turn reduce their sending rate, and so on recursively until reaching the emitter. This mechanism effectively provides end-to-end congestion control *via* point-to-point congestion control provided by TCP, while flow control is discussed at the application layer. Finally, as a background thread, a eTor relay is in charge of maintaining connections alive, including retrying broken connections.

The use of multiple guard relays by the client enables to leverage multipathing, and sending data over multiple paths, thereby enabling a higher overall capacity of the `IOC`

1. KIST [122] has shown that considering the kernel space queue helps reducing delays and improving reactivity. We limit our implementation to user-space queues to ease development, especially as file transfer is tolerant to high latency.

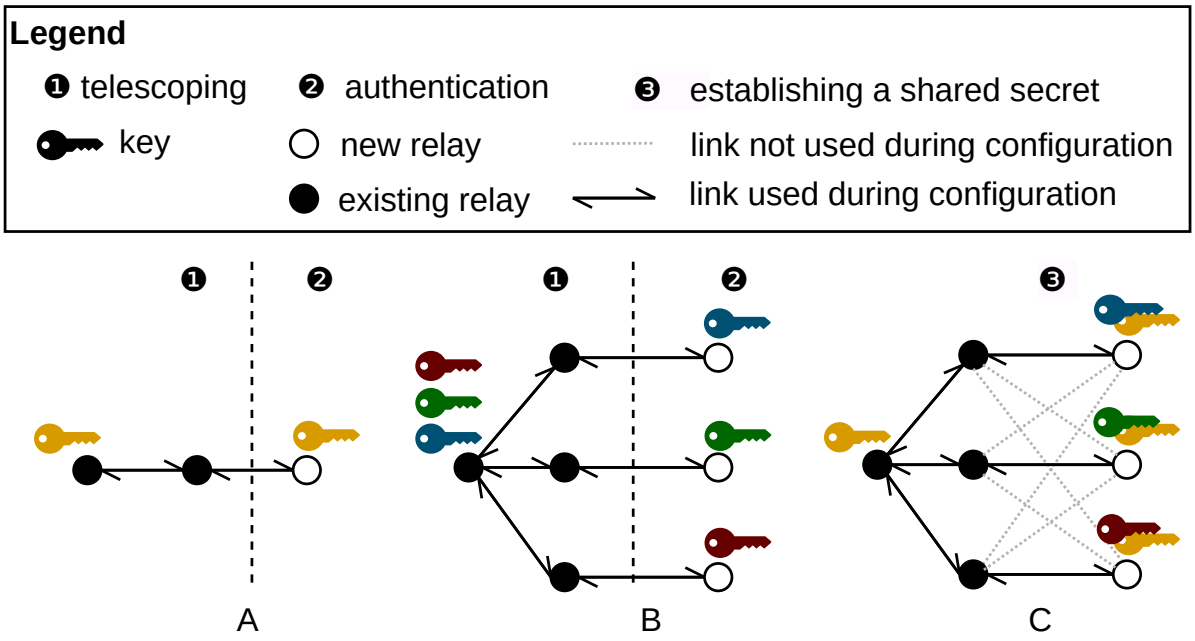

Figure 4.3 – Creation of an IOC in eTor.

compared to a single Tor circuit build over the same relays.

In the following, we discuss how multipath circuits' lifetime is handled, from their creation to their deletion.

4.3.3 Lifetime of an IOC

The proper management of an IOC lifetime has an impact on its robustness and performance, as well as on its security properties. Choices made at the creation of the circuit, but also during maintenance operations (e.g., when one of the path gets temporarily unavailable in the IOC) all impact these properties. We present in this section how eTor preserves Tor's forward secrecy and sender-receiver anonymity properties when establishing IOCs. We also discuss how we separate the concerns of connection closing and circuit destruction thanks to a garbage collection approach.

Preserving Forward Secrecy. The opening of an IOC is based on Tor's circuit establishment mechanism and hence inherits from both (i) its Telescoping Mechanism (TM) [58], and (ii) its Authentication Protocol (TAP) [**goldberg2006security**, 58]. IOCs are built incrementally: the part of an IOC that has been already configured is used to configure its next relays. For instance, and as depicted in Figure 4.3 A, Tor circuits are built in

two phases. In phase ❶, the initiator leverages the TM protocol to inform the first hop (that has already been configured) that the circuit will be extended with a second hop. In phase ❷, the TAP protocol is used to configure the new relay and add it to the circuit.

To maintain forward secrecy (i.e., negotiating a session key from a long-lived public cryptography key pair), eTor reuses both the TM and TAP protocols, but with multiple relays at once, according to the number of relays per hop. However, in eTor, *relay cells* may be addressed to any relay of an hop requiring, therefore, to build circuits in 3 phases; the additional phase allowing to setup a shared secret for relays belonging to the same hop. For instance, in Figure 4.3 B, the initiator in eTor runs the two Tor phases 3 times, i.e., as many times as there are relays per hop, as there is a different key for each relay. Finally, in Figure 4.3 C, phase ❸ takes place: beforehand the shared secret is encrypted with the key of each target relay to preserve the session key's security properties. The resulting cipher is then sent over the circuit under construction to each relay, enabling cells to be sent to any relay of this hop.

Sender-receiver anonymity. Tor's onion services, used to connect two participants anonymously, are slightly modified to allow communication over `IOCs`. Both the `introduction` and `rendez-vous` points, which are extremities of a circuit, are now made of multiple relays. As a result, (i) descriptors registered in the Onion Service directory must be updated accordingly, and (ii) `rendez-vous` points sent through introduction points must now be a set of relay addresses, and not anymore to a single relay address.

Garbage collection. In contrast with Tor, the loss of an individual relay-to-relay connection in an `IOC` does not result in the loss of the complete end-to-end circuit. Each such individual connection is, instead, periodically retried by relays at a given layer to account for temporary unavailabilities of one failed next-hop relay. Relays use an exponential back off strategy to avoid saturating the network with unnecessary connection attempts. Edge relays do not expect clients to gracefully close `IOCs`: they rely instead on a garbage collection mechanism. Relays automatically discard entries for `IOCs` that have not conveyed data for a configured duration, and associated connections to next-hop layer are also terminated.

We have detailed thus far how multipath `IOC` are managed, assuming that participants know how to generate circuits identifiers I, keys K and select relays R_j^i. While I and K are simply randomly generated, the selection of relays follows a more complex logic that aims at maximizing the `IOC` availability based on prediction figures, and that we detail next.

4.3.4 Selecting Relays for Multipath Circuits

`IOCs` are composed of relays chosen by the extremities of the circuit (communicating participants). eTor uses an algorithm that packs n relays per-hop while ensuring that the hop availability probability $P(H)$ (i.e., the probability that at least one relay will be active to transfer a cell to the next layer) remains above a certain threshold ϵ: $P(H) \geq 1 - \epsilon$.

The probability of hop availability is time-dependent, thus computed at a time t and is valid for an interval h. Its estimation is derived from relay availability estimation $P(R)$ on the same period given by the relay directory. We compute the probability of at least one relay remaining up at any given time in equation 4.1.

$$P^{[t,t+h]}(H) = 1 - \prod_{j=0}^{n} 1 - P^{[t,t+h]}(R_j) \tag{4.1}$$

When selecting the relays for a hop, we start by picking n random relays then compute the resulting availability for this hop. If the computed availability is below the defined threshold (defined through ϵ), the relay with the lowest availability is replaced by a new randomly picked relay. This operation is repeated until a hop exceeds or equals the threshold availability.

One can ask how many relays risk being discarded as this can influence network security, the worst-case being all relays having the same availability. The rationale is that a high availability relay will not compensate for the presence of ones with low availability. We derive a higher bound on relay availability based on this fact, knowing that all relays above this bound will never be evicted from a hop. To compute our bound, we suppose that all relays have the same availability $P(R)$ and find the minimum availability to meet the n and ϵ criteria, thus we seek $P(R)$ for $P(H) = 1 - \epsilon$, results are described in equation 4.2.

$$\begin{aligned} 1 - \epsilon &= 1 - (1 - P(R))^n \\ P(R) &= 1 - \sqrt[n]{\epsilon} \end{aligned} \tag{4.2}$$

As we can see, as long as $P(R) \geq 1 - \sqrt[n]{\epsilon}$, the relay will never be ignored. Knowing all relays in the network and their availability enables to deduce the subset that will be never above the threshold, giving also an upper bound to compute anonymity quantification. This result can also be used to configure ϵ and n by a network administrator, in order to find a trade-off that will maximize performances and security.

Indeed, depending on the hop for which they are selected, relays may be picked for

different lifespans. New standard hops will be provisioned for each circuit, we arbitrarily estimate the lifetime of a circuit to one hour, giving us $h_s = 1$ supposing h is in hours. However, the first hop of a participant must be fixed between available paths in order to protect from predecessor attacks. We are aligning our choice on Tor, provisioning the first hop for 4 months, hence $h_g = 2880$. We refer to this first hop as a guard hop. Relay directories must then announce two predictions for each relay: $R^{[t,t+h_g]}$ and $R^{[t,t+h_s]}$.

Computation of Relays Availability Probabilities. Relays must register with the relay directory periodically. Relay directory nodes can, therefore, log the periods of availability and unavailability of each relay. This data can be used to feed predictors able to guess the future availability of relays based on their past behavior. Availability information is, in fact, already used by the relay directory in Tor to assign the "guard" flag to relays that, among other criteria, require a minimal availability profile. Instead of just publishing a flag for each edge relay, eTor extends the information returned by the relay directory with prediction results for the two considered horizons (1 hour and 4 months). It is then up to the client to decide on the suitability of a given relay for a hop in its `IOC` based on this information and using the previously-described selection algorithm.

The choice of the predictor used to guess the expected availability from past availability information received from each edge relay is left to the decision of the designers of the relay directory. This allows flexibility, e.g., to adapt from the use of existing Tor edge relays to include future domestic edge relays. In our evaluation, we use an exponentially weighted moving average (EWMA) predictor and show its effectiveness on the trace of availability for existing Tor edge relays.

We are now ready to conclude the description of eTor with the presentation of flow control mechanisms over established `IOCs`, and the use of the circuits to perform the exchange of files.

4.3.5 Reliable Transmission and Flow Control

A difference between regular Tor circuits and `IOCs` is that the latter, unlike the former, does not respect directly the in-order delivery semantics of TCP. Indeed, while in a Tor circuit each inter-relay link uses TCP and deliver messages in order, resulting in in-order delivery between communicating endpoints, in an `IOC` different packets for the same connection may take different routes and arrive, therefore, out-of-order at their destination. Some packets may even be lost due to relay crashing after acknowledging the packet re-

ception to the previous layer (preventing replay by the latter). The in-order delivery in eTor is enforced at the client library level.

The flow control and reliable transmission mechanism we use is the Selective Repeat Automatic Repeat-reQuest (ARQ) [214, 215, 216] algorithm, a sliding-window protocol that lets the sender send several chunks at once, and allows the receiver to accept them out of order. ARQ performs flow control similarly to Tor's window, preventing from overloading relays with data. Each chunk is associated with its position (or ID). The sender sends the chunk and its ID in an onion cell using the `IOC`, while the receiver sends back an acknowledgment (ACK) with the same ID for each received data chunk in a relay cell again, on the same `IOC`. When the sender does not receive an ACK after sending a chunk, it retries sending it after a timeout of several seconds. The exchange completes once all chunks have been acknowledged. We implement file transfer directly on top of ARQ, by offering a TCP-socket abstraction as for the regular Tor circuits.

4.4 Security Analysis

In this section, we analyze the security of eTor. We start by reviewing known attacks against onion routing networks, and then quantify the end-to-end anonymity that the system provides.

4.4.1 Attacks

A variety of attacks have been documented against onion routing networks and in particular against Tor. We start by reviewing these attacks systematically and assess how (and to which extent) they apply to eTor, using the existing Tor network as a comparison point.

Correlation Attacks

For `IOC`, we need to adapt the Tor rule to transient edge relays, and we pick a set of relays (instead of a single one), that is also fixed for 4 months. In Section 4.4.2, we show how picking multiple entry guards is counterbalanced with a bigger network in terms of anonymity.

Epistemic Attacks

Alternative designs have been studied to reduce the scalability cost of the relay discovery while providing protection against epistemic attacks [90, 89, 88, 91]. A large-scale

deployment of eTor should leverage one of this network discovery solution to fulfill its needs for network discovery while maintaining scalability.

Users Exposition As we seek to make users contribute with their personal equipment (domestic edge relays), publishing relay information raises new questions on the possibility they offer to infer users' behavior. eTor's design does not involve collecting more relay information than Tor already does, that is, mainly the relay IP address and its availability every hour. Incidentally, we built our dataset in Section 4.5 by collecting such data.

In contrast with Tor core relays, however, the availability of domestic edge relays is much more likely to be correlated with users' behavior than servers running in data centers. The question is not simply about data collection because, as long as relay IP addresses are being advertised, many information could be inferred, such as user's sleep patterns, by simply querying the relay directory. If an attacker learns someone's IP address, it can infer whether or not it is a eTor infrastructure contributor and obtain its availability history (by probing it if not provided by the directory) that might correlate with the user's sleep patterns for example.

It is however possible to mitigate these risks. First, by having a directory that publishes only a subset of the relays to each user as discussed in the previous paragraph, an attacker would be able to collect only partial information. Second, by advising users to avoid associating their IP address with their identity for parties they do not trust. Third, by encouraging users to run eTor on edge devices that are less correlated with their behaviors, such as NAS, routers, and workstations. Finally, the directory could publish a relay descriptor only if it is not too distinctive from others, e.g. by requiring that other relays with similar availability patterns and in the same IP address range be registered. To put it in the nutshell, the end-user has multiple choices to limit or hide its availability patterns if required and given the network reaches a critical mass, it is also possible to proactively filter relays that could expose their owners due to their uniqueness.

4.4.2 End-to-end Anonymity Quantification

Our goal is now to estimate what anonymity can a user expect from our security goals and the studied existing attacks. We want especially to quantify the impact of an attacker running compromised relays–or being able to observe them–on the network. As we consider multiple sessions, we consider the anonymity provided by entry guards.

First, we consider an attacker that wants to know if two users are communicating

together. Such an attacker would need to break *Relationship Anonymity*. We refer to such an event as D_1. Attacker $\mathcal{A}$ would need to corrupt both receiver and sender entry guards, resulting in equation 4.3.

$$P_{Tor}[D_1] = (P_{Tor}[\text{pick } \mathcal{A} \text{ relays}])^2 \approx \frac{N^2_{\mathcal{A} \text{ relays}}}{N^2_{\text{Tor relays}}} \tag{4.3}$$

With eTor, we have not one but n relays per hop. Following the Tor approach, we pick a conservative approach: one packet transiting via a corrupted relay is enough to de-anonymize a user, thus picking one attacker relay in the entry hop is enough to conduct all attacks. The probability of breaking *Relationship Anonymity* with eTor is then presented in equation 4.4.

$$\begin{aligned} P_{\text{eTor}}[D_1] &= (n \cdot P_{\text{eTor}}[\text{pick } \mathcal{A} \text{ relays}])^2 \\ &\approx n^2 \cdot \frac{N^2_{\mathcal{A} \text{ relays}}}{N^2_{\text{eTor relays}}} \end{aligned} \tag{4.4}$$

An attacker $\mathcal{A}$ might only be interested in knowing if a targeted user sent a file in the network, whomever the recipient is, or could be one of the participants of the exchange. These two scenarios break our two other security features: *Sender Unlinkability* and *Sender-Receiver Anonymity*. We refer to these two events as D_2 as they require both to break only one entry guard to success. We present the adapted equations 4.5 and 4.6.

$$P_{Tor}[D_2] = P_{Tor}[\text{pick } \mathcal{A} \text{ guard}] \approx \frac{N_{\mathcal{A} \text{ relays}}}{N_{\text{Tor relays}}} \tag{4.5}$$

$$\begin{aligned} P_{\text{eTor}}[D_2] &= n \cdot P_{\text{eTor}}[\text{pick } \mathcal{A} \text{ relays}] \\ &\approx n \cdot \frac{N_{\mathcal{A} \text{ relays}}}{N_{\text{eTor relays}}} \end{aligned} \tag{4.6}$$

To compare the security difference between Tor and eTor, we introduce an indicator α. If $\alpha = 1$, the attacker success probability in both systems is identical, if $\alpha > 1$, eTor is harder to attack than Tor, if $\alpha < 1$, eTor is easier to attack than Tor. We focus our analysis on D_2 as it is the least advantageous for eTor, which gives us equation 4.7.

$$\alpha = \frac{P_{Tor}[D_2]}{P_{\text{eTor}}[D_2]} \approx \frac{N_{\text{eTor relays}}}{n \cdot N_{\text{Tor relays}}} \tag{4.7}$$

If we consider networks of the same size, then α will be lower than one and Tor will be more advantageous. If we seek the equilibrium, the eTor network will need to be n

times bigger than the Tor one: the increased number of entry guards will be compensated by a bigger network. Finally, if the eTor considered network is even bigger than n times the Tor one, then eTor provides better anonymity probability against the studied de-anonymization attacks.

Still, before comparing both networks, we must estimate the difference between the *considered* networks by the guard selection algorithm and the *entire* network as known by the relay directory. Indeed, in practice, both Tor and eTor have mechanisms to prevent low availability relays from being picked, reducing the relay pool to less than the one advertised by the relay directory. In Tor, only relays with the "guard" flags are considered. In eTor, all relays are considered but relays below a certain threshold will have a lesser probability of being taken as they may be discarded if the final hop availability is too low. To simplify our analysis, and based on the upper bound derived in the approach (equation 4.2), we consider that they will not be selected at all. In other words, we consider only relays whose $P(R) \geq 1 - \sqrt[n]{\epsilon}$. We update the previous estimation to obtain the final α in equation 4.8.

$$\alpha = \frac{P_{Tor}[D_2]}{P_{\text{eTor}}[D_2]} \approx \frac{|R \in \text{eTor}, P(R) \geq 1 - \sqrt[n]{\epsilon}|}{n \cdot N_{\text{Tor guards}}} \tag{4.8}$$

We argue that in practice we could easily obtain an $\alpha > 1$ considering the current Tor user/relay base. Currently, Tor features around $\sim 3,000$ guards over its $\sim 6,500$ available relays at a given point in time. For eTor, we need to set both n and ϵ parameters while considering an availability prediction dataset to obtain a value for the considered network size. Anticipating our evaluation, we leverage a dataset of edge relay availability built from the Tor consensus (as seen in Figure 4.1), and keep two eTor configurations: $n = 2, \epsilon = 0.01$ (2 relays/hop, 99% availability/hop) and $n = 3, \epsilon = 0.0001$ (3 relays/hop, 99.99% availability/hop). These two configurations enable us to keep respectively 66% and 40% of our edge devices fleet. If the 2M Tor users would run an eTor relay, it would result in, respectively, a 1.3M and 800k relays considered network, providing an α of 220 and 88 respectively, hence greatly improving anonymity. Considering that not all anonymity networks users will run a relay, we argue that the ratio remains higher than 1, thus increasing anonymity, even if only a fraction of the users run a relay. If we consider that only 10% of users run a relay, the α ratio drops to 22 and 8 but remains way higher than 1. Decreasing the value to 1% of the users brings us around the equilibrium point with α of 2.2 and 0.88: requesting a 99.99% per-hop availability provides worse anonymity compared to Tor while targeting a smaller value of 99% can still benefit the users.

We conclude that starting from the point where 1% of Tor users would accept to run a eTor relay, it is possible to provide anonymity similar to Tor with eTor's design, with obviously a much greater overall network capacity. Passed this value, it is even preferable to use eTor instead of Tor.

4.5 Evaluation

To evaluate eTor effectiveness against realistic data, we leverage the dataset introduced in Section 4.2 to simulate volatility. We then use this dataset to highlight eTor main strengths: high availability and throughput despite churn. In the last part, we discuss eTor protocol parameters' impact on anonymity quantification and predictors soundness in light of our dataset.

4.5.1 Building an Edge Volatility Dataset

Tor consensus discussed in Section 4.2 is published every hour. Each consensus contains the whole list of relays, including their availability and IP address. By collecting them for one year, we were able to retrace relays history, especially their churn. As eTor targets durable edge devices, we keep in our test datasets only relays whose IP address is tagged as residential and features a lifetime higher than one week, resulting in a sample of 2522 relays containing their availability hour per hour as seen by the Tor relay directory. This sample is the same one as the one described in Section 4.2 and depicted as the "Edge" category in Figure 4.1. Excluding our microbenchmark, all our other results are based on this dataset.

4.5.2 Performance & Comparison

eTor fights churn both at the macro and micro levels. At the macro level, eTor must select relays to build circuits such as a path of available relays between participants exists at any time inside the circuit. At the micro-level, eTor must dynamically route cells inside the multipath circuit, ensuring good network usage while overcoming faults. We evaluate both of these approaches independently.

First, to evaluate the effectiveness of eTor at the macro level, we simulate what we call a user journey to encompass the fact that the guard hop is chosen once every 4 months

and then will be used for all circuits. We want to be sure that a user will then be able to build working circuits during the whole guard lifetime.

In Figure 4.4, we compare the ratio of successful circuits for 1000 users against three configurations of eTor and our ground-truth during the 4 months lifespan of a guard. Our ground truth uses a very naive approach by building single path circuits with random relays. Our three eTor configurations differ from the desired availability per hop: 1%, 0.1% and 0.01%. Following analysis conducted in following section 4.5.3, we picked the number of relays per-hop that provide the best anonymity according to our dataset (2 relays for 1% and 0.1%, 3 for 0.01%).

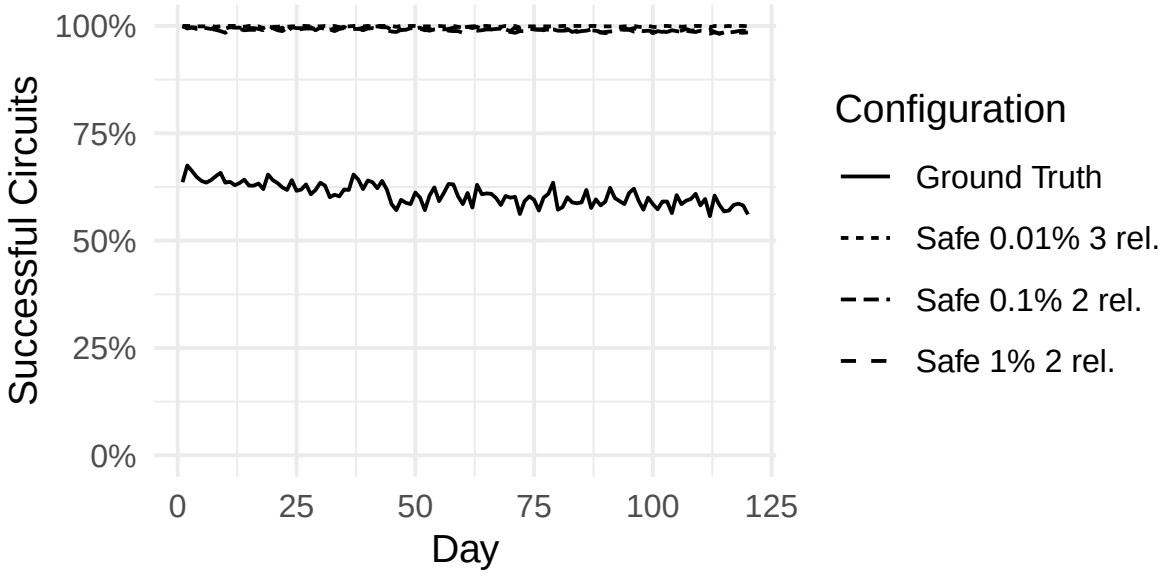

Figure 4.4 – During a guard lifetime, average ratio of working circuits for 1000 users.

With 33% of failures, we confirm that the simple approach of our ground truth is unsatisfactory with edge devices. At the same time, we show that eTor relay availability predictions combined with its multi-relay per-hop circuit scheme enable to build circuits that work close to 100% of the time during the whole life of a guard. Independently of the used algorithm, we notice the stability of circuits success ratio over time, encompassing that despite their volatility, edge devices feature interesting properties like volatility stability over time and availability independence between them.

Now that we have demonstrated that eTor can build working multipath circuits across edge devices, we show that eTor can reliably and efficiently route onion cells through these circuits. To effectively evaluate our whole system, including our onion pro-

tocol, our cell routing, and our file exchange protocol, we built a real-world application. We implemented the whole software in 2 000 lines of Scheme, leveraging the `libsodium` library for the cryptography primitives, and the Linux `epoll` interface to handle the network. We then conducted a file exchange of 2GB between two clients involving 12 relays (3 relays per hop in a 4 hops circuit). To assess our protocol network effectiveness, we limited the bandwidth per relay to 30 Mbit/s by using the Linux `tc` tool. Furthermore, we simulated two unavailability periods of one minute on the same hop, the first one involving the loss of one node while the second features the loss of two nodes. We plot the effective bandwidth experienced by users during the file transfer in Figure 4.5.

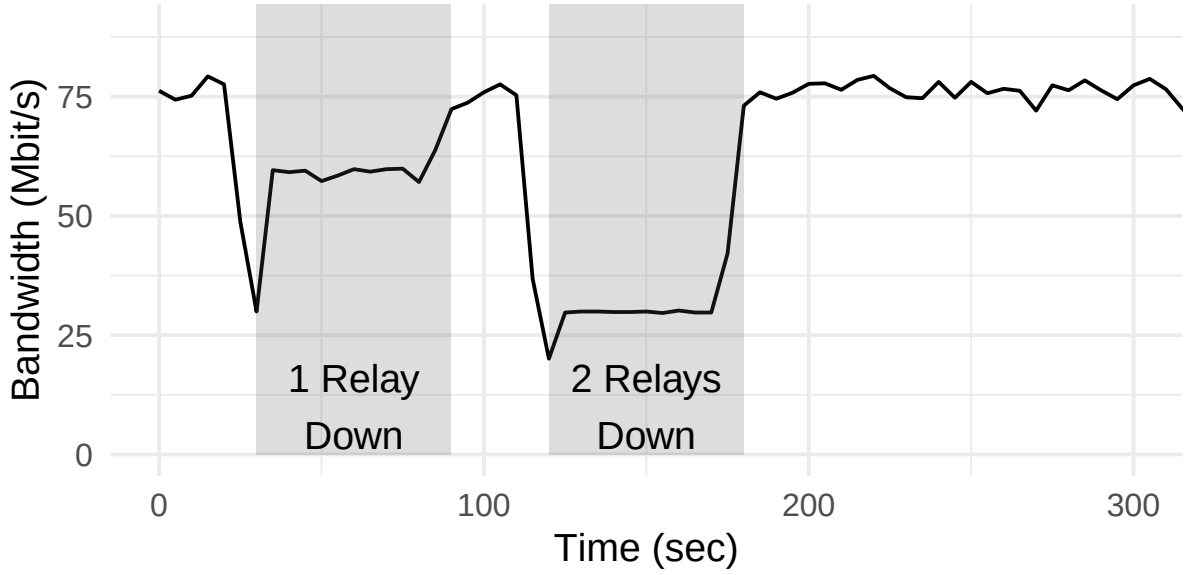

Figure 4.5 – Transfer bandwidth as experienced by end users over eTor with 30 Mbit/s relays

First, we see that despite two unavailability events, eTor adapts instantaneously to the conditions without failing the transfer. When the relay goes down, the retransmission of lost packets is automatically triggered by a timeout, and the traffic is automatically load-balanced to the remaining available relays. When offline relays go back online, the transfer is again immediately rebalanced between all the available relays. Such load balancing is made possible thanks to our per onion cell routing decision. Additionally, no eTor handshake is required to start again sending traffic to the relay as circuits are not bound to connections.

Furthermore, our protocol, thanks to its design decisions, can support high throughput

both by being lightweight (eg. using symmetric key cryptography), and inducing few network overheads (eg. using circuits and storing routing information in relay memory). While we argue our prototype demonstrates the benefits of eTor design, we think the maximum throughput could be increased to the line rate (from 75 Mbit/s to close to 90 Mbit/s) with a more advanced implementation.

4.5.3 Anonymity Sensitivity Analysis

eTor has two network configuration options: a maximum hop unavailability per hop, and a number of relays per hop. Increasing the first one will reduce the number of considered candidates, thus the considered size of the network for the anonymity quantification, while increasing the second one increases the chance of an attacker to be one of the guards. In this section, we depart from the upper bound given in the approach and theoretical equations provided in the security analysis to take an empirical approach. We base our study on sampling: we build more than 100k circuits with our algorithm for each point and look at how many time an attacker (with a 100% availability) is present in the built hop.

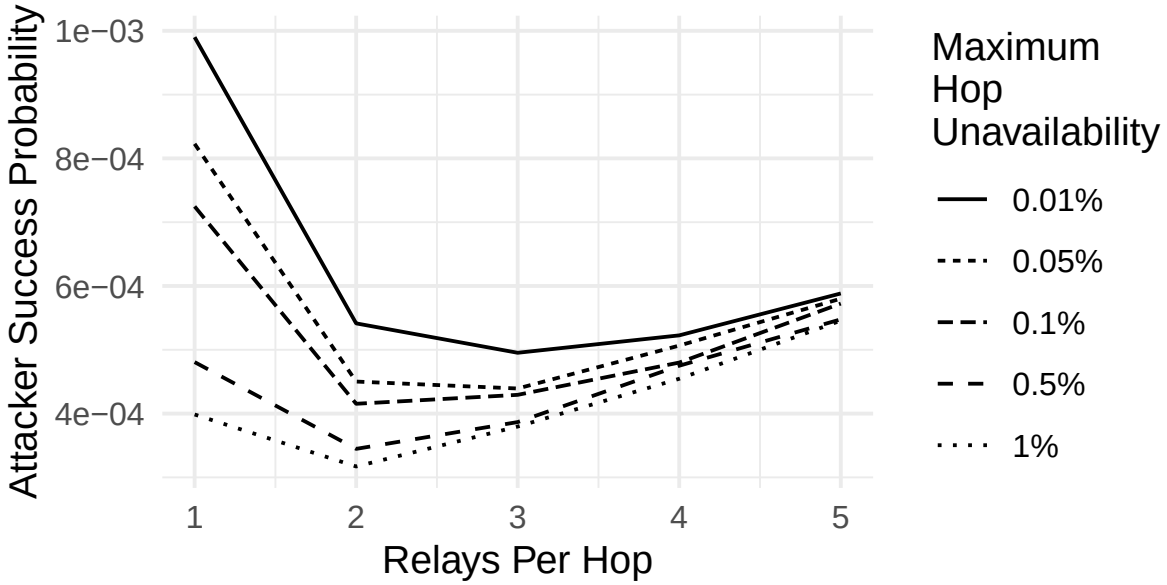

Figure 4.6 – Influence of the number of relays per hop on the anonymity quantification for a fixed availability target.

In Figure 4.6, we look at the tension between the relays per-hop and an attacker's success probability. Given our availability target and our edge dataset, we observe an

inflection point between 2 and 3 relays per hops according to the desired availability per hop. In other words, for a hop availability target between 99% and 99.9%, it is wiser to use a guard of 2 relays instead of 1. If a user targets an availability of 99.99%, it is then even preferable to use 3 relays per hop. The rationale behind these results is, if we pick only one relay per hop, we will exclude all relays that have less than 99% (or 99.9% or 99.99%) of availability, in other words, most of the edge relays. By picking two relays per hop, it is possible to reach the same hop availability with lower per relay availability, thus enlarging our considered network which finally results in better anonymity.

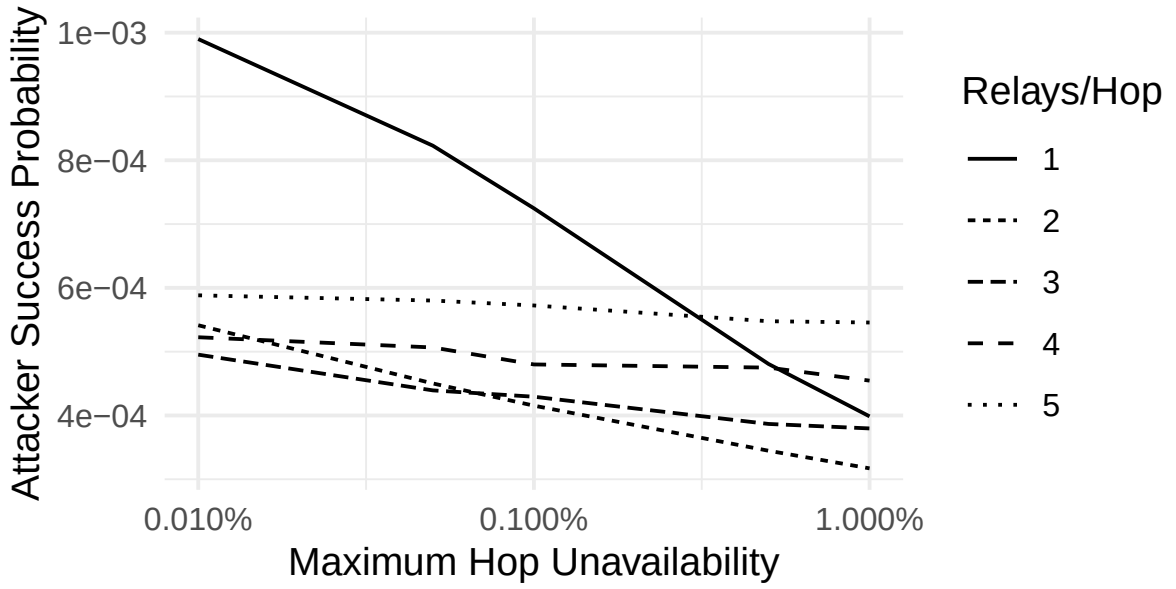

Figure 4.7 – Influence of the targeted availability ϵ on the anonymity quantification for a fixed number of hop.

Figure 4.7 presents the same data seen from the perspective of target hop availability. It highlights the fact that availability has a cost, especially with one relay per-hop where an attacker's success rate increases quickly with availability.

We conclude that with edge devices, with a target availability superior or equal to 99%, it is more interesting to consider, in terms of anonymity, at least two relays per guard instead of one. We also note that picking one relay per guard can quickly increase an attacker's success probability by aggressively reducing the number of considered relays.

4.5.4 Predictors Soundness

To ensure high availability hops with as few relays as possible, we use predictions to evict relays that have too low availability. It is possible as edge devices observed volatility in the Tor consensus seems to be very predictable, featuring repetitive patterns. As proof, we evaluated three basic predictors: Unit, EWMA, and a Markov Chain plus a random one. We recall that we have two types of hops: standard and guards, each has a different prediction horizon (4 months and 1 hour respectively).

Unit is the most simple predictor, it simply returns the value of the previous timestep availability. EWMA is a weighted moving average that we study with two decay parameters: 0.9 and 0.99. Finally, we built a two-state Markov chain model (available or unavailable) and updated transitions again with EWMA with the two same decay parameters.

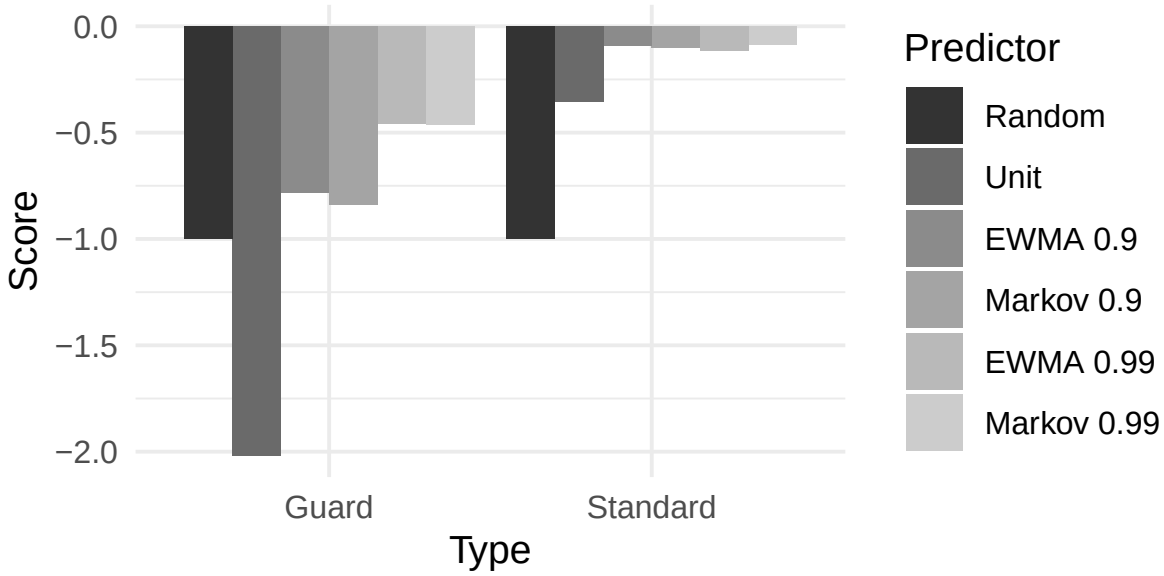

Figure 4.8 – Evaluation of the effectiveness of 3 predictors: random, EWMA and markov chain. We use a logarithm scoring rules, close to zero is better. Guard and Standard differs from their prediction horizon: 4 months and 1 hours respectively.

Figure 4.8 presents score results for all predictors (closer to zero is better). Unsurprisingly, it is easier to predict relay availability on a short horizon than on a longer one. Standard relays availability can be predicted with high precision, highlighting scores superior to -0.1 for all EWMA and Markov predictors. Still, it still makes sense to predict guard availability: EWMA with a 0.99 decay features a score of -0.5 which is far superior

from not predicting, which is equivalent to a random predictor and its score of -1.

In conclusion, eTor builds robust multipath circuits by combining multiple relays per-hop and predictions. eTor fully leverages its circuits by providing fast adaptations to change and high throughput.

4.6 Conclusion

We presented eTor, a proposal for an extension of the Tor anonymity network providing increased capacity by leveraging relays at the edge of the network. Until now, edge relays in Tor were generally disregarded, and domestic-grade hardware contributed by users disregarded, due to their volatility and limited resources. We argue that leveraging these devices, while taming their imperfect availability and capacity, is key in enabling Tor to scale and support more demanding (i.e., bandwidth-intensive) applications.

We introduced in eTor the novel concept of an *Intertwined Onion Circuit* (`IOC`). `IOCs` use multipathing, and can thus forward data as long as one relay per hop remains available. Combined with relay availability prediction, eTor is able to build `IOC` with availability close to 100%. Inside an `IOC`, eTor can deliver high user-facing bandwidth, achieving an effective 75 Mbit/s over 30Mbit/s relays by spreading data through available relays in the `IOC` during a data transfer. eTor features flow control and reliable transmission support in order to mask faults and unavailability inside the `IOC` from the user application.

The use of `IOCs` does not come at the cost of a lower anonymity. We have shown that if at least of 1% of current Tor users were to run an edge relay, the success probability of a de-anonymization attack would be inferior to that of Tor despite of the fact that we pick more relays per circuit.

From now, we focused our interest on one-to-one communication by improving their latency and their throughput. In the following, we shift our focus to how to enable efficient group communication.

Chapter 5

Paving the Way to Decentralized Anonymous Group Communication

Most group communication over the internet is mediated by third party servers. This design is often chosen as it is easy to deploy and manage thanks to the full control on the servers it confers to the application provider. Furthermore, it enables basic privacy as users can't see each others' identity (like IP address).

However, from a privacy standpoint this design has some severe drawbacks. Anonymity between users is already provided by the network. Introducing a mediating server that can collect data or metadata (even with end-to-end encryption activated) would reduce privacy by enabling operators to build knowledge about their users' behaviours. From an economic and performance perspective, this design is also not adapted to community-based anonymity networks: it requires some parties to maintain and pay for expensive servers. We conclude that alternatives must be explored for community-based anonymity networks, such as Tor, that both preserve privacy and better fit to community-based organization.

In the last decade, gossip protocols, also known as epidemic protocols, have been widely adopted as a key functional building block to build distributed systems. For instance, gossip protocols have been used for overlay construction [217, 218], membership and failure detection [219, 220, 221, 222], aggregating data [223], and live streaming [224], [225]. This wide adoption comes from the resilience, simplicity, and natural distribution of gossip protocols [226, 219, 227]. These properties are particularly adapted to community-based networks where nodes are volatile and low powered. Furthermore, they do not require third party infrastructure.

Gossip-based dissemination can be simply represented as the random phone-call problem; at the beginning, someone learns of a rumor, and calls a set of random friends to propagate it. As soon as someone learns of a new rumor, in turn, she randomly propagates it to her own set of friends, and so on recursively. Further, depending on whether there

are one or more sources of rumors (i.e. dissemination of messages from one or multiple nodes), gossip protocols may be either single or multi source. In both cases, randomness and recursive probabilistic exchanges provide scalability, robustness and fault tolerance under high churn to disseminate data while staying simple.

However, due to its probabilistic aspect, gossip-based dissemination implies high redundancy with nodes receiving the same message several times. Many algorithms have been studied to limit the number of exchanged messages to disseminate data, using different combinations of approaches such as *push* (a node can push a message it knows to its neighbors), *pull* (a node pulls a messages it does not know from its neighbors) or *push-pull* (a mix of both) for either single- or multi-source gossip protocols [149][150][69].

In this chapter, we make a significant step beyond these protocols, and provide better performance with respect to the state of the art of multi-source gossip protocols.

The key principle of our approach is to consider redundancy as a key advantage rather than as a shortcoming by leveraging Random Linear Network Coding (RNLC) to provide efficient multi-source gossip-based dissemination. Indeed, it has been shown that RLNC improves the theoretical stopping time, i.e., the number of rounds until completing the protocol, by sending linear combinations of several messages instead of a given plain message, which increases the probability of propagating something new to recipients [156, 157].

Unfortunately, applying RNLC to multi-source gossip protocols is not without issues, and three key challenges remain open. First, existing approaches suppose that a vector, where each index identifies a message with its associated coefficient as a value, is disseminated. This approach implies a small namespace. In the context of multi source, the only option is to choose a random identifier over a sufficiently large namespace to have a negligible collision probability. However this does not scale. Some algorithms provide a mechanism to rename messages to a smaller namespace [228], but this kind of technique is not applicable to gossip protocols as it would substantially increase the number of exchanged messages, and inherently the delay. Second, to reduce the complexity of the decoding process, messages are split in groups named generations. Existing rules to create generations require having only one sender, which is impractical in the context of multiple sources. Third, the use of RNLC implies linear combinations of multiple messages. This leads to potential partial knowledge of received messages, making precise message pull requests useless and breaks pull-frequency adjustments based on missing-message counts.

In this chapter, we introduce CHEPIN, a CHeaper EPidemic dissemINation approach

for multi-source gossip protocols. To the best of our knowledge, our approach is the first one to apply RNLC to multi-source gossip protocols, while overcoming all the inherent challenges involved by the use of network-coding techniques.

More precisely, we make the following contributions.

— We solve the identifier namespace size *via* the use of sparse vectors. Additional headers sent over the network represent 10% or less of the total message size.

— We create generations for messages from multiple sources by leveraging Lamport timestamps. All messages sharing the same clock are in the same generation whatever their source.

— We overcome the issue of partial message knowledge by providing an adaptation of push, pull, push-pull gossip protocols. We pull specific generations instead of specific messages.

— We introduce updated algorithms to make our approach applicable to the current state of the art of multi-source gossip protocols.

— Finally, we evaluate CHEPIN thoroughly by simulation. We show that our solution reduces bandwidth overhead by 25% and delivery delay by 18% with respect to Pulp [69], while keeping the same properties.

5.1 Network Coding Background

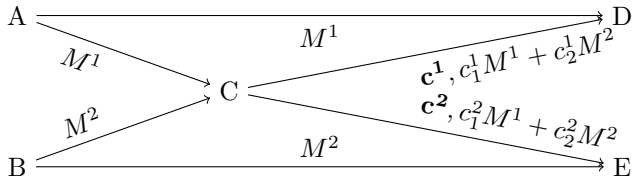

Figure 5.1 – With RLNC, C can send useful information to D and E without knowing what they have received

RLNC [229] provides a way to combine different messages on a network to improve their dissemination speed by increasing the chance that receiving nodes learn something new. In Figure 5.1, node C cannot know what D and E have received. By sending a linear combination of M^1 and M^2, nodes D and E can respectively recover M^2 and M^1 with the help of the plain message they also received. Without RLNC, node C would have to send both M^1 and M^2 to D and E involving two more messages. Every message must have the same size, defined as L bits thereafter. To handle messages of different size, it is possible

to split or pad the message to have a final size of L bits. The message content has to be split as symbols over a field $\mathbb{F}_{2^n}$.

Encoded messages are linear combinations over $\mathbb{F}_{2^n}$ of multiple messages. This linear combination is not a concatenation: if the original messages are of size L, the encoded messages will be of size L too. An encoded message carries a part of the information of all the original messages, but not enough to recover any original message. After receiving enough encoded messages, the original messages will be decodable.

To perform the encoding, the sources must know n original messages defined as $M^1, ..., M^n$. Each time a source wants to create an encoded message, it randomly chooses a sequence of coefficients $c_1, ..., c_n$, and computes the encoded message X as follows: $X = \sum_{i=1}^{n} c_i M^i$. An encoded message thus consists of a sequence of coefficients and the encoded information: (c, X).

Every participating node can recursively encode new messages from the one they received, including messages that have not been decoded. A node that received $(c^1, X^1), ..., (c^m, X^m)$ encoded messages, can encode a new message (c', X') encoded by choosing a random set of coefficients $d_1, ..., d_m$, computing the new encoded information $X' = \sum_{j=1}^{m} d_j X^j$ and computing the new sequence of coefficients $c'_i = \sum_{j=1}^{m} d_j c_i^j$.

An original message M^i can be considered as an encoded message by creating a coefficient vector $0, ..., 1, .., 0$ where 1 is at the ith position. The encoding of a message can therefore be considered as a subset of the recursive encoding technique.

Even if there is no theoretical limit on the number n of messages that can be encoded together, there are two reasons to limit it. First, Gauss-Jordan elimination has an $O(n^3)$ complexity, which becomes rapidly too expensive to compute. Then, the more messages are encoded together, the bigger the sequence of coefficients while the encoded information remains stable. In extreme cases this can result in sending mainly coefficients on the network instead of information. To encode more data, splitting messages in groups named generations solves the previous problems, as only messages in the same generation are encoded together. However applying network coding to epidemic dissemination raises several challenges.

Assigning a message to a generation Generations often consist of integers attached to messages. Messages with the same generation value are considered in the same generation and can be encoded together. The values must be assigned in such a way that enough messages are in the same generation to benefit from the RLNC properties but

not too many to keep the decoding complexity sufficiently low and to limit the size of the coefficients sent on the network. In a single-source scenario, the size of the generation is a parameter of the protocol, and is determined by counting the number of messages sent in a given generation. However, with multiple sources, there is no way to know how many messages have been sent in a given generation.

Sending coefficients on the network Coefficients are generally sent under the form of a dense vector over the network. Each value in the vector is linked to a message. In a single-source scenario, this is not a problem: the source knows in advance how many messages it will send and can assign each message a position in the vector and start creating random linear combinations. In the case of multiple sources, the number of messages is not available, and waiting to have enough messages to create a generation could delay message delivery and above all, the network traffic required to order the messages in the dense vector would ruin the benefits of network coding.

Pulling with RLNC When doing pull-based rumor mongering, a node must have a way to ask what rumors it needs. Without network coding, it simply sends a message identifier to ask for a message. But sending a message identifier in the case of network coding raises several questions: does the node answer only if it has decoded the message? Or if it can generate a linear combination containing this message?

Estimating the number of missing packets Some gossip protocols need to estimate the number of missing messages. Without network coding, the number of missing packets corresponds to the number of missing messages. But with RLNC, it is possible to have some linear combinations for a given set of messages but without being able to decode them. All the messages are considered as missing but one packet could be enough to decode everything.

5.2 Contribution

5.2.1 System model

Our model consists of a network of n nodes that all run the same program. These nodes communicate over a unicast unreliable and fully connected medium, such as UDP over the Internet. Nodes can join and leave at any moment, as no graceful leave is needed;

crashes are handled like departures. We consider that each node can obtain the addresses of some other nodes in the system via a Random Peer Sampling service [230]. There is no central authority to coordinates nodes: all operations are fully decentralized and all exchanges are asynchronous. We use the term message to denote the payload that must be disseminated to every node of the network, and the term packet to denote the exchanged content between two nodes. We consider the exchange of multiple messages and any node in the network can inject messages into the network, without any prior coordination between nodes (messages are not pre-labeled).

5.2.2 Solving RLNC challenges

In our multiple-independent-source model and with our goal to cover push and pull protocols, we propose new solutions to the previously stated challenges.

Assigning generations with Lamport timestamps With multiple senders, we need to find a rule that is applicable based only on the knowledge of a single node when assigning a generation as relying on the network would be costly, slow and unreliable. We chose Lamport timestamps [231] to delimit generations by grouping every message with the same clock in the same generation. This method doesn't involve sending new packets as the clock is piggybacked on every network-coded packet. When a node wants to disseminate a message, it appends its local clock to the packet and updates it, when it receives a message, it uses its local clock and the message clock to update its clock. This efficiently associates messages that are disseminated at the same time to the same generation.

Sending coefficients in sparse vectors When multiple nodes can send independent messages, they have no information about which identifiers are assigned by the other nodes. Consequently, they can only rely on their local knowledge to choose their identifiers. Choosing identifiers randomly on a namespace where all possible identifiers are used would lead to conflicts. That is why we decided to use a bigger namespace, where conflict probabilities are negligible when identifiers are chosen randomly. On a namespace of this size, it is impossible to send a dense vector over the network, but we can send a sparse vector: instead of sending a vector $c^1, ..., c^n$, we send m tuples, corresponding to the known messages, containing the message id and their coefficient: $(id(M^{i_1}), c^{i_1}), ..., (id(M^{i_m}), c^{i_m})$.

Pulling generations instead of messages A node sends a list of generations that it has not fully decoded to its neighbors. The target node answers with one of the generation it knows. To determine if the information will be redundant, there is no other solution than asking the target node to generate a linear combination and try to add it to the node's local matrix. During our tests, we observed that blindly asking for generations did not increase the number of redundant packets compared to a traditional Push-Pull algorithm asking for a list of message identifiers, while greatly decreasing message sizes.

Count needed independent linear combinations To provide adaptiveness, some protocols need to estimate the number of useful packets needed to receive all the missing messages. Without RLNC, the number of needed packets corresponds to the number of missing messages. With RLNC, partially decoded packets are also considered as missing messages, but to decode them we need fewer packets than missing messages. In this case, the number of required useful packets corresponds to the number of required independent linear combinations.

5.2.3 CHEPIN

To ease the integration of RLNC in gossip-based dissemination algorithms, we encapsulated some common logic in Algorithm 1. We represent a network-coded packet by a triplet: $\langle g, c, e \rangle$, where g is the generation number, c an ordered set containing the network-coding coefficients and e the encoded payload.

We define 3 global sets: *rcv*, *ids*, and *dlv*. *rcv* contains a list of network-coded packets, as described before, which are modified each time a new one is received to stay linearly independent until all messages are decoded. *ids* contains a list of known message identifiers under the form $\langle g, gid \rangle$ where g is the generation and *gid* is the identifier of the message inside the generation. By using this tuple as a unique identifier, we can reduce the number of bytes of *gid* as the probability of collision inside a generation is lower than the one in the whole system. Finally *dlv* contains a list of message identifiers similar to *ids*, but contains only the identifiers of the decoded messages.

The presented procedure relies on some primitives. RANK returns the rank of the generation, by counting the number of packets associated to the given generation. SOLVE returns a new list of packets after applying a Gaussian elimination on the given generation and removing redundant packets. DELIVER is called to notify a node of a message (if the same message is received multiple times, it is delivered only once).

Algorithm 1 Process RLNC Packets

```
1:  g ← 0                                                    ▷ Encoding generation
2:  rcv ← Set()                                              ▷ Received packets
3:  ids ← OrderedSet()                                       ▷ Known message identifiers
4:  dlv ← OrderedSet()                                       ▷ Delivered message identifiers

5:  function ProcessPacket(p)
6:      if p = ∅ then
7:          return False

8:      ⟨g¹, c¹, _⟩ ← p
9:      oldRank ← Rank(g¹, rcv)
10:     rcv ← Solve(g¹, rcv ∪ {p})

11:     if oldRank = Rank(g¹, rcv) then
12:         return False                                     ▷ Packet was useless

13:     for all ⟨id, _⟩ ∈ c¹ do
14:         ids ← ids ∪ {⟨g¹, id⟩}                           ▷ Register new identifiers

15:     for all ⟨g², c², e⟩ ∈ rcv do
16:         ⟨id, _⟩ ← c²[0]
17:         if g¹ = g² ∧ len(c²) = 1 ∧ ⟨g², id⟩ ∉ dlv then
18:             dlv ← dlv ∪ {⟨g², id⟩}
19:             Deliver(e)                                   ▷ New decoded message

20:     if g¹ > g ∨ Rank(g, rcv) ≥ 1 then
21:         g ← max(g¹, g) + 1                               ▷ Update Lamport Clock
22:     return True                                          ▷ Packet was useful
```

This procedure updates the 3 previously defined global sets, delivers decoded messages and returns the usefulness of the given packet. Internally, the node starts by adding the packet to the matrix and by doing a Gaussian elimination on the packet's generation (line 10), if decoding the packet does not increase the matrix rank, the packet is deemed useless and the processing stops here. Otherwise, the node must add unknown message identifiers from the packet-coefficient list to the known-identifier set. After that, the node delivers all the messages decoded thanks to the received packet and stores their identifiers in *dlv*. Finally, the node checks if the clock must be updated.

Algorithms 2 and 3 show how the above procedures can be used to implement push and pull gossip protocols. For push, we do not directly forward the received packet, but instead forward a linear combination of the received packet's generation after adding it to our received-packet list. For Pull, we request generations instead of messages. Like existing protocols, we keep a *rotation* variable that rotates the set of missing identifiers, allowing missing generations to be generated in a different order on the next execution of the code block.

5.3 Application to Pulp

To apply our network-coding approach to a concrete use case, we design CHEPIN-Pulp, a protocol inspired by Pulp [69]. Pulp achieves cost-effective dissemination by optimizing the combination of push-based and pull-based gossip. In particular, nodes disseminate each message through a push phase with little redundancy due to a fanout and a TTL configured to reach only a small portion of the network. As the push phase does not provide a complete dissemination, the message will be retrieved by the rest of the network during the pull phase. To this end, each node periodically sends its list of missing messages to a random node. The target node answers with the first message it knows. To discover missing messages, nodes piggyback the list of recently received messages on every packet exchange. To improve reactivity and reduce delays, Pulp provides a pull-frequency adaptation mechanism based on the node's estimations of the number of missing messages and of the usefulness of its pull requests.

On top of the two previously defined Algorithms 2 and 3, we propose a push-pull algorithm inspired by Pulp: Algorithm 4. First, we must convert Pulp's message-discovery mechanism to RLNC. This part consists of exchanging recent message histories via a trading window. The trading window is generated by the GetHeaders function, and

Algorithm 2 Push

```
1: k, ittl ← ...                                   ▷ Push fanout and initial TTL
2: dottl, dodie ← ...                              ▷ Push strategy

3: function SendToNeighbourgs(h, headers)
4:     for k times do
5:         p ← Recode(h, rcv)
6:         Send(Push, p, headers)

7: function Broadcast(m, headers)
8:     id ← UniqueID()
9:     p ← ⟨g, {⟨id, 1⟩}, m⟩
10:    dlv ← dlv ∪ {⟨g, id⟩}
11:    ProcessPacket(p)
12:    headers.ttl ← ittl
13:    SendToNeighbourgs(g, headers)

14: function NCPush(p, headers)
15:    ⟨h, _, _⟩ ← p
16:    if ProcessPacket(p) ∨ ¬dodie then
17:        if dottl ∧ headers.ttl ≤ 0 then
18:            return
19:        if dottl then
20:            headers.ttl ← headers.ttl − 1
21:        SendToNeighbourgs(h, headers)
```

Algorithm 3 Pull

```
1: rotation ← 0                                              ▷ Rotation position

2: function NCPullThread(headers)
3:     ask ← OrderedSet()
4:     rotation ← rotation + 1  mod |ids \ dlv|
5:     for all m ∈ Rotate(ids \ dlv, rotation) do
6:         ask ← ask ∪ Gen(m, rcv)
7:     Send(Pull, ask, headers)

8: function NCPull(asked, headers)
9:     p ← ∅
10:    if ∃ g ∈ asked, Rank(g, rcv) > 0 then
11:        p ← Recode(g, rcv)
12:    Send(PullReply, p, headers)

13: function NCPullReply(p)
14:    ProcessPacket(p)
```

is added to every packet. Upon reception, the trading window is retrieved and its new identifiers are added to the *ids* set. The major difference with Pulp is that we do not trade identifiers of delivered messages but any identifiers we know of, even if we compare both approaches in Section 5.4.

The adaptation mechanism is the second feature of Pulp, the pull frequency is adapted according to the number of missing packets and the usefulness of the pull requests. Our only modification is made on how to compute the number of missing packets, as we retain the number of needed independent linear combinations instead of the number of missing messages. To do so, we compute the difference between the number of message identifiers and the number of independent linear combinations we have.

5.4 Evaluation

We evaluated our solution in the Omnet++ simulator, using traces from PlanetLab and Overnet to simulate respectively latency and churn [1].

To assess the effectiveness of CHEPIN, we implemented a modified version of Pulp

1. Code is accessible at https://gitlab.inria.fr/WIDE/chepin/flexnet

Algorithm 4 CHEPIN-Pulp

1: $ts, sm \leftarrow ...$ ▷ Trading window size and margin
2: $\Delta_{adjust}, \Delta_{pull_{min}}, \Delta_{pull_{max}} \leftarrow ...$ ▷ Periods config.
3: $dottl, dodie \leftarrow True, True$ ▷ Set push strategyy
4: $\Delta_{pull} \leftarrow \Delta_{adjust}$

5: **function** GetHeaders
6: $start \leftarrow \text{MAX}(0, |ids| - sm - tm)$
7: $end \leftarrow \text{MAX}(0, |ids| - sm)$
8: **return** $\{tw : ids[start : end]\}$

9: **upon receive** Push$(p, headers)$
10: $ids \leftarrow ids \cup headers.tw$
11: NCPush$(p,$ GetHeaders$())$
12: **upon receive** Pull$(asked, headers)$
13: $ids \leftarrow ids \cup headers.tw$
14: NCPull$(asked,$ GetHeaders$())$
15: **upon receive** PullReply$(p, headers)$
16: $ids \leftarrow ids \cup headers.tw$
17: NCPullReply(p)

18: **thread** every Δ_{pull}
19: NCPullThread(GetHeaders())
20: **thread** every Δ_{adjust}
21: $missingSize \leftarrow |ids| - |rcv|$
22: **if** $missingSize > prevMissingSize$ **then**
23: $\Delta_{pull} = \frac{\Delta_{adjust}}{missingSize - prevMissingSize + prev_{useful}}$
24: **else if** $missingSize > 0 \wedge prev_{useless} \leq prev_{useful}$ **then**
25: $\Delta_{pull} \leftarrow \Delta_{pull} \times 0.9$
26: **else**
27: $\Delta_{pull} \leftarrow \Delta_{pull} \times 1.1$
28: $\Delta_{pull} \leftarrow \text{MAX}(\Delta_{pull}, \Delta_{pull_{min}})$
29: $\Delta_{pull} \leftarrow \text{MIN}(\Delta_{pull}, \Delta_{pull_{max}})$
30: $prev_{useless}, prev_{useful} \leftarrow 0$
31: $prevMissingSize \leftarrow missingSize$

as described in Section 5.3, and compare it with the original Pulp protocol.

5.4.1 Experimental setup

Our experiments consist in disseminating 1 000 messages at a rate of 150 messages per second, with each message being emitted by a different node amongst the 1 000 nodes in the network. We note that at this rate, per-node network coding would induce important delays, as 150 msg/sec represents less than one message emitted every 6 seconds per node.

Every node can communicate with any other node in the network. Each message weighs 1KB and has a unique identifier. Δ_{adapt} is set to 125 ms. We consider that the latency difference that might be induced by the additional headers is negligible with respect to the payload size and the considered latency. The whole RLNC algorithm [2] was run during the simulation, including the Gaussian Elimination part, but encoding/decoding was limited to the coefficients and the first 2 bytes of the payload. With the previous parameters, a simulation runs on a single core of an Intel i7-7600U CPU at 2.80GHz in less than 2 minutes using less than 400MB of RAM.

In order to accurately simulate latency, we use a PlanetLab trace [232]. Latency averages 147 ms with a maximum of almost 3 seconds. Most of the values (5th percentile and 95th percentile) are between 20 ms and 325 ms. Finally, we have a long tail of values between 325 ms and the maximum value.

5.4.2 Configuring the Protocols

To configure the protocols, we chose an experimental approach. First, we selected a suitable value for the size of the trading window. As explained in Section 5.3, too small values of this parameter result in wasted pull requests, and missing messages, while too large ones lead to wasted bandwidth. We therefore tested the ability of the original Pulp, and of our solution to achieve complete dissemination (i.e. all messages reach all nodes) with different trading window sizes, and a safety margin of 10. Results, not shown for space reasons, show that our solutions reaches complete dissemination with trading window sizes of at least 6, while Pulp requires trading-window sizes of at least 9. For the rest of our analysis, we therefore considered a trading-window size of 9, and a safety margin of 10. Nonetheless, this first experiment already hints at the better efficiency of our network-coding-based solution.

2. Code is accessible at https://gitlab.inria.fr/WIDE/chepin/libflexcode/

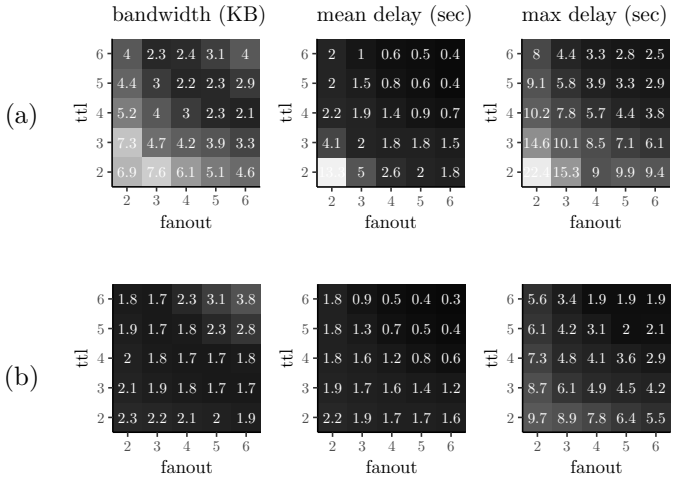

Figure 5.2 – Pulp (5.2a) and CHEPIN-Pulp (5.2b) behavior under various configurations of the protocol (fanout and time to live)

Next, we selected values for fanout and TTL. Figure 5.2 reports the delivery delays and bandwidth consumption of the two protocols with several values of these two parameters. Each measurement has been done 32 times. Only the average is reported; the measured values are no more than +/- 5% of the average value for the bandwidth and the minimum delay, and no more than +/- 60% of the average value for the maximum delay. To measure bandwidth consumption, we consider the ratio between the average amount of bandwidth consumed by the protocol, and the lower bound represented by the bandwidth required for the same task in a tree structure in a stable network. First, we observe that in terms of delays and bandwidth used, our network-coding variant is more stable than the original Pulp. That is, with low values of fanout and TTL, the original algorithm deteriorates faster.

Next, we see that our network-coding variant performs better or similarly for every combination of fanout and TTL both in terms of sent bandwidth and delay. The best configuration in term of sent data for Pulp corresponds to the configuration $k = 6, TTL = 4$ with 2.12 KB for 1KB of useful data and an average of 0.67 seconds to disseminate a message. Our network-coding solution reduces delay to 0.55 s, with a bandwidth consumption of 1.83KB/msg. With a fanout of 5, our solution further decreases the consumed bandwidth to 1.66 KB/msg but with a slight increase in delay (0.83 s). Clearly, to achieve the minimum delays, the best strategy consists in boosting the push phase by increasing the TTL, but this defeats the bandwidth-saving goal of Pulp and our approach. As a

result, we use the configuration with $k = 6, TTL = 4$ for both protocols in the rest of our comparison.

5.4.3 Bandwidth and delay comparison

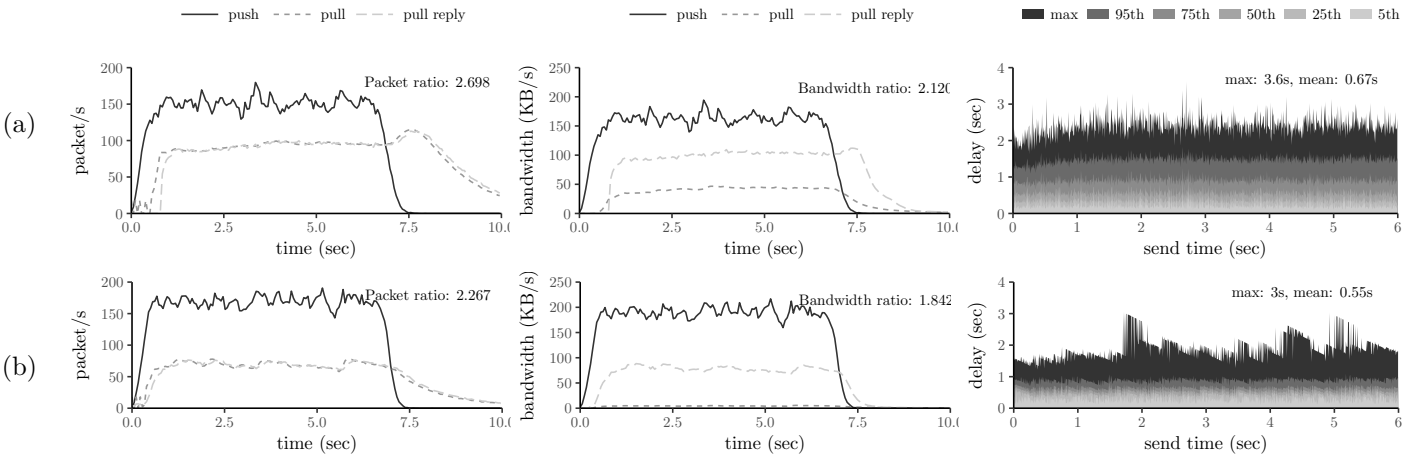

Figure 5.3 – Comparison of the exchanged packet rate, used bandwidth and message delay for 5.3a Pulp and 5.3b CHEPIN-Pulp

We evaluate how our algorithm performs over time in Figure 5.3. First, we logged the number of packets sent per second for the three types of packets: push, pull and pull reply. As we configured the two protocols with the same fanout and TTL, we would expect to see almost the same number of push packets. But our network-coded variant sends 12% more push packets. Pulp stops forwarding a push packet if the corresponding message is already known. But since our variant can use a large number of linear combinations, our algorithm manages to exploit the push phase better, thereby reducing the number of packets sent in the pull phase: 33% fewer pull and pull reply packets. This strategy enables us to have a packet ratio of only 2.27 instead of 2.70.

As network coded packets include a sparse vector containing message identifiers and values, CHEPIN-Pulp has larger pull and pull reply packets than Pulp. Considering push packets, we also send more of them, which explains why we send 17% more data for these packets. At the same time, CHEPIN-Pulp reduces the header part of pull messages by asking for generations (groups of message) instead of messages while reducing the chances of redundancy in replies thanks to RLNC. More generally, the pull phase is shorter due to a more efficient push phase. These two facts enable us to have a data ratio 1.84 instead

2.12.

Finally, we study the distribution delay of each message. As our algorithm has a longer push phase, delays are shorter on average. We see a downward slope pattern on our algorithm's delays, especially on the maximum-delay part. This pattern can be explained by the fact that decoding occurs at the end of each generation, so messages that are sent earlier wait for longer than the most recent ones.

5.4.4 Adaptiveness optimization

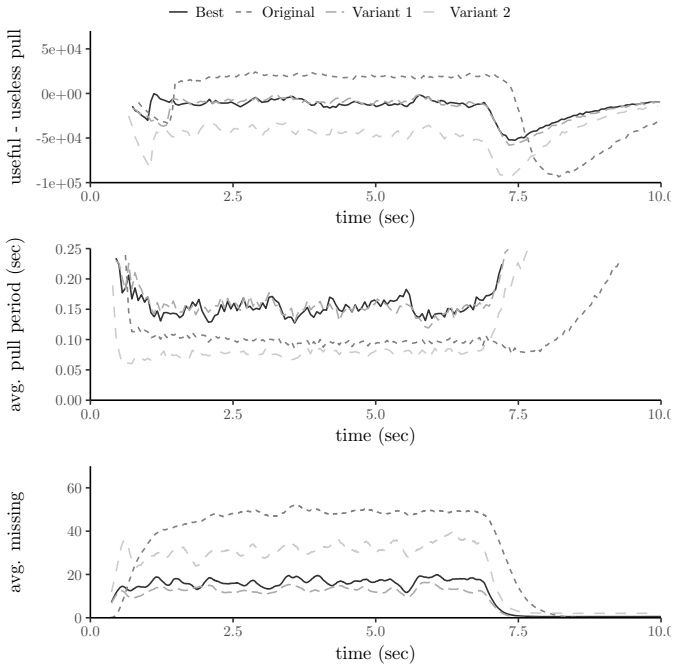

Figure 5.4 – How adaptiveness algorithms impact the protocol efficiency

We now carry out a sensitivity analysis to understand the reasons for our improved performance. To this end, Figure 5.4 compares CHEPIN-Pulp identified as Best with Pulp and with two intermediate variants.

The first variant corresponds to a modification in the GetTradingWindow function of Algorithm 1. Instead of using the message identifiers contained in the *ids* variable, we use the message identifiers contained in the *dlv* variable like in the case of the standard Pulp protocol. In other words, we disseminate the identifiers of messages we have decoded and not those we are aware of.

The second variant is a modification of how we count the number of missing messages at line 21 in Algorithm 4. For this variant, we do $missingSize \leftarrow |missing|$ like in the original Pulp. We thus evaluate the number of missing messages by counting all the message identifiers we have not yet decoded, without taking into account the progress of the decoding in our generations.

The two variants perform worse than our solution both in terms of delay and bandwidth. Variant 1 does not manage to achieve complete dissemination with a fanout of 6 and a TTL of 4, while Variant 2 achieves complete dissemination but at a higher cost ratio: 2.40 instead of 1.83 for our solution. This shows the importance of the modifications we made to the Pulp protocol.

In Figure 5.4, we see that Pulp has a better ratio of useful over useless messages, a smaller pull period and more missing messages than CHEPIN-Pulp due to having more messages to pull, caused by a less efficient push phase. Best, Variant 1 and Variant 2 have the same push phase, and consequently the same number of messages to pull. We see that the pull strategy of Variant 2 is not efficient: it asks for many messages more frequently with a smaller useful-over-useless ratio. Variant 1 performs similarly to Best, but not better. Moreover its per-disseminated-message efficiency is lower as it does not provide complete dissemination.

5.4.5 Behaviour under network variation

Figure 5.5 shows how our algorithm performs under different network configurations and gives the average generation size for each point. We observe that the difference between CHEPIN-Pulp and Pulp is greater with larger message rates, longer latency, and larger networks, while being correlated with generation sizes. For a given latency distribution, larger generations tend to lead to more efficient coding thereby improving performance. In particular, generations become larger when there are more messages being disseminated in the network at approximately the same time as is the case when increasing latency, message rate, or network size.

At one extreme, when we have only one message per generation, we have the same model as Pulp: asking for a list of generation identifiers is similar to asking for a list of message identifiers in Pulp. At the other extreme, the risk is to have too many messages per generation, impacting the node's local resources. However, we see that the generation size increases logarithmically, as when we have more messages per generation, we also have more messages to disseminate the knowledge of this generation.

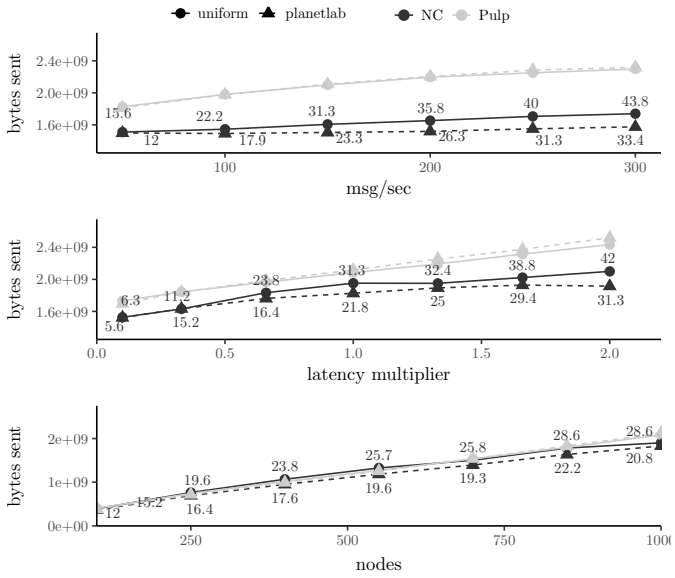

Figure 5.5 – Variation of network configuration. The number near each point indicates the average generation size for NC.

Figure 5.5 also displays the results obtained with a uniform latency distribution with a lower bound set to the 5^{th} percentile of the PlanetLab dataset and an average similar to the one from the PlanetLab dataset, resulting in an upper bound of 274ms. We observe that CHEPIN-Pulp's improvement over Pulp is greater with the PlanetLab distribution, even if this means smaller generations.

We use an Overnet trace to simulate churn[233]. The trace contains more than 600 active nodes over a total of 900 with continuous churn—around 0.14143 events per second.

We use this trace replayed at different speeds to evaluate the impact of churn on the delivery delays of our messages, as plotted on Figure 5.6. Specifically, we consider three speeds: 500, 1000 and 2000 times faster for respectively 71, 141 and 283 churn events per second. We see that the original Pulp algorithm is not affected by churn, as the average and maximum delivery delays stay stable and similar to those without churn. Considering the average delay, it's also the case for our algorithm: the average delay does not evolve. The maximum delay does not evolve significantly either. However we can see huge differences in the shape of the maximum delay for each individual message. Indeed, the decoding order and the generation delimitation are affected by churn, but this has limited impact on message dissemination.

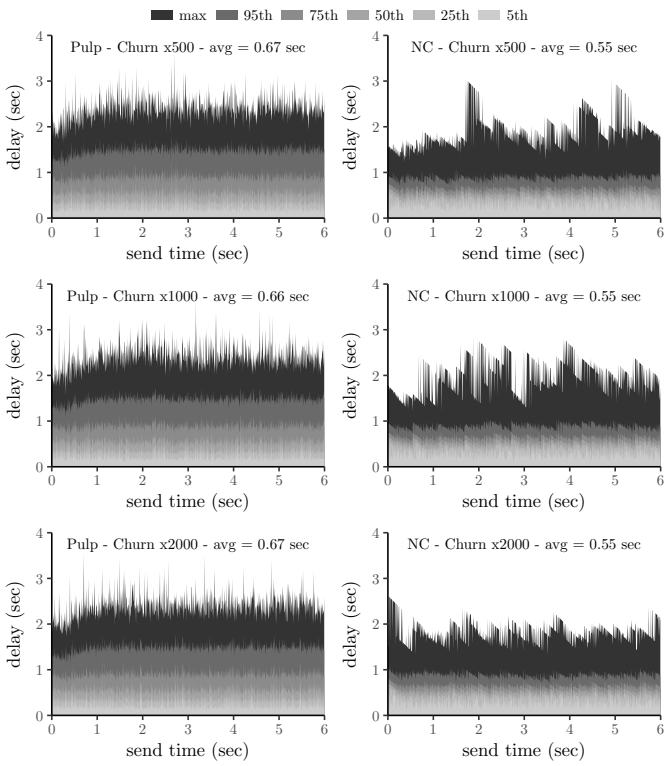

Figure 5.6 – Delay observation with churn on the system

5.5 Conclusion

Due to their properties and resistance to failure, we think data dissemination protocols are particularly adapted to group communication over a community-based anonymity network such as Tor. They could work over Tor's hidden service infrastructure, solving the NAT and firewall problem, while overcoming circuit breakage and member disconnection without relying on a third-party server. However, these advantages come at the cost of bandwidth spent by the embedded redundancy in the protocol. We introduced CHEPIN, a multi source gossip protocol that uses Lamport clocks to create generations, and sparse vectors to exchange coefficients. We demonstrated that it is possible to apply RLNC for push and pull algorithms by thoroughly evaluating our approach. At worst, CHEPIN performs like the state of the art. At best, CHEPIN significantly improves both the delay and bandwidth consumption. As future work, we would like to investigate the benefits of overlapping generations on message discovery and efficiency. We are also interested by improving CHEPIN's adaptiveness and extending its generation management.

This work led to the following publication: Yérom-David Bromberg, Quentin Dufour,

and Davide Frey, « Multisource Rumor Spreading with Network Coding », *in*: *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, IEEE, 2019, pp. 2359–2367 [68].

In the following, we analyze the synergies that could emerge from our three contributions Donar, eTor and CHEPIN before concluding on our work.

Chapter 6

Future Work

In a goal to enable users to communicate in a private manner, we contributed Donar to enable Voice over IP communication, eTor to make file transfers a first class citizen by introducing *organic scaling* and CHEPIN to provide a reliable and efficient group communication protocol we think would integrate well with Tor's onion services. Still, to create a real paradigm shift in our relation with our communication tools, a lot of work remains. In the following, we explore some future work that could leverage these contributions to help build technologies and a society that values anonymity.

6.1 Onion Routing

Stabilize even more latency With Donar, we were able to stabilize latency enough to have a real-time protocol, VoIP, working seamlessly. However, we know that even outside Donar, people encounter problems with VoIP. These problems have many sources, ranging from internet peering to WiFi including the "last mile" link (ADSL, coaxial, etc). In some cases, our current algorithm directly addresses these problems. For example, thanks to its probing, Donar will take the path having the least peering problems. When Donar's algorithm has no impact on the latency source, like when the user has a bad WiFi link, performance improvements on Donar can still help to compensate for other latency sources. First, we think that our scheduling algorithm can be refined, both by using more advanced logic and doing a larger evaluation. Second, by taking a white-box approach: we referenced a large number of latency sources (relay rate limiting, relay prioritization, etc.) that we could use to create a local model and better predict latency.

From traditional multipath to multipath onion circuits While Donar was built on legacy Tor with real time communication in mind, eTor focuses on organic scaling on a standalone implementation: the two systems are very different and we think that synergies exist between the eTor and Donar designs. As we better understand the benefits of

Multipath Onion Circuits (MOC), we could work on the minimal set of modifications to port our contribution on Tor. With this patched version of Tor, we could then analyze how DONAR scheduling could be merged into eTOR's design. We identified two main possibilities to integrate DONAR's contributions: (i) by enhancing the end-to-end scheduler with the MOC design or (ii) by moving the scheduler to relays as MOC enables relays to take routing decisions inside them. As there is no obvious better solutions to us, we would compare both of them in various environments with different levels of congestion and churn. Our longstanding goal is to better understand what is the full range of benefits that MOC can offer by enabling relays to take routing decisions, especially compared to traditional multipath.

6.2 Group communication

Resist malicious peers CHEPIN is particularly adapted to groups where members heavily trust each other. Still, CHEPIN could also benefit from more "open" usage, where everyone is invited to participate. In this case, the protocol must be able to resist malicious behaviors; we note especially Sybil attacks [234], where the attacker floods the entire network with its profiles and Eclipse attacks [235], where the attacker circles a target with its profile. In this context, a byzantine-resistant RPS [236] has already been proposed and it could be interesting to study how CHEPIN could be adapted to such untrusted environments.

CHEPIN in the wild CHEPIN demonstrated its effectiveness compared to state of the art but it still must be manually configured according to its environment. We reduced the impact of configuration but to seek wider adoption, we think that an adaptive algorithm is a necessity. We envision two main directions: (i) introducing coordination servers and (ii) dropping generations. One or more coordination servers, possibly moving and untrusted, could feed adapted parameters to participants by observing (part) of the traffic. The other option is to drop parameters, especially generations, by changing the algorithm. More specifically, we could leverage the work on our flexible matrices to drop Lamport-timestamp-based generations and replace them with opportunistic encoding and decoding. Opportunistic decoding of packets would be done in two times: first with the backlog of already decoded messages, then the packet is put in the single flexible decoding matrix on which we apply one round of Gaussian Elimination. If a packet is decoded, it is

removed from the matrix. Opportunistic encoding of a packet would consist of encoding the new packet either with the current decoding matrix, or the recent backlog, or a yet to be defined mix of both. With this more versatile solution, we could then study how to integrate it with Onion Services and evaluate its performances in this context. We think of large group chat, collaboration tools or more demanding usage like group VoIP.

6.3 Long-term goals

Reconsider client/server protocols We observed that Internet usage often involves connecting people. For convenience, people are not directly connected but mediated through third party servers. By focusing its design on being compatible with existing applications and protocols, Tor also adopted this design. Today, we can observe that most discussions on Onion Services occurs through forums or email also maintained by third parties.

During this PhD thesis, we observed that Onion Services solved many of the inconveniences that led to the introduction of these third parties. Onion Services are not bound to the infrastructure, not limited in number, preserve anonymity including location, bypass technical limitations such as NAT or firewalls, support authentication, etc. We conclude that many protocols could be revisited to abandon the need for third party servers and be directly built on top of the primitives offered by Onion Services.

The most brilliant example of this integration is TorChat. An identity is bound to an Onion Service, being cryptography backed it also serves as authentication and encryption. Availability can be inferred by the presence of the Onion Service in the Tor DHT. Each user can have as many identities as she wants to separate the different part of her life. Communication is done directly without the need for any third party server. Similarly, with Donar, we observed how easy it was to make a VoIP call without requiring any server or specific configuration from our users.

Instead of adapting applications independently, we argue that it would be more efficient to adopt a systematic approach. Our work would constitute in referencing Onion Service primitives, identifying patterns, proposing building blocks on top of Onion Service primitives in order to build a global framework. This framework could be applied to help freeing existing protocols and applications, like VoIP, chat and collaboration tools, from their centralization.

Chapter 7

Conclusion

We conducted our work following the observation that many privacy issues take their root in the traces we leave. Based on our observation, we tried to understand why privacy enhancing technologies, especially anonymity networks, were not used yet by people to better control their traces. We observed that while a large number of anonymity networks have been designed, only a more restrained set has been effectively deployed, and none of them support the wide variety of modern communication. To help users adopt anonymity networks, we contributed solutions to widen their use.

Donar brings real-time communication through carefully scheduled multipath. Compared to existing multipath approaches, Donar is built with anonymity network specificities in mind and evaluated in real conditions. As a result, we were able to make real calls while meeting industry standards in terms of quality of service.

eTor introduces "organic scaling" to support bandwidth intensive communication. Running relays at the edge, behind a residential connection on non dedicated hardware, is often discouraged over Tor, as it results in lower user experience. With eTor, it is no longer the case as our contribution captures the notion of availability at its heart, provisions redundancy accordingly and seamlessly adapts to relay availability change.

CHEPIN enables efficient group communication without a coordinating third party. Abandoning the third party meant that we had to provide services while overcoming natural client churn. Such a property is given by gossip protocols, that naturally embed redundancy but require to send more messages. Introducing network coding makes it possible to reduce the number of exchanged messages while keeping the benefits of gossip protocols. We adapted theoretical algorithms, that were designed with non-realistic assumptions, to a real world system. In our evaluation, we were able to demonstrate that we were improving on many characteristics at once: latency, bandwidth and sensitivity to parameters.

To put it in a nutshell, we enlarged the communication types supported by anonymity networks to enable real-time, high-bandwidth and group communication. Combined with the possibilities offered by onion services, we think it can enable us to build more direct communication, both reducing the number of generated traces and who can see them. We hope that democratizing privacy enhancing technologies, in conjunction with other fields of research including humanities, could help reduce privacy issues and result in fairer communication among humans.

Appendix A

Résumé en français

En 2014, Cambridge Analytica a collecté secrètement des données de millions d'utilisateurs de Facebook pour construire leur profil psychologique. Grâce à ces données, ils avaient pour objectifs d'influencer les élections aux États-Unis en 2016. Si l'impact réel de leur campagne sur la population n'est pas clair, il a été montré qu'il est en effet possible de déterminer notre profil psychologique à travers notre comportement sur Facebook. Une fois ces informations rendues publiques, de nombreux citoyens se sont mobilisés au point que le PDG de Facebook a du se justifier devant le congrès américain. Loin d'être un évènement isolé, nous avons observé que le scandale Cambridge Analytica s'inscrit dans un ensemble plus large de controverses liées à la vie privée.

Le concept de vie privée a été abordé par de nombreux universitaires et ne fait pas totalement consensus sur sa définition. Dans cette thèse nous retenons la définition inspirée de Benett qui dit que la vie privée est une approche pour réglementer l'analyse des données personnelles par les organisations publiques et privées ainsi qu'une façon d'empêcher l'observation excessive des comportements humains. Sans en être son antagoniste, les intérets des acteurs pratiquant une forme de surveillance, au sens de l'observation, de l'analyse et de modification du comportement humains, ont un effet négatif sur la vie privée des individus. Face à cette tension entre surveillance et vie privée, nous nous intéressons aux méthodes qui permettent aux individus de mieux contrôler quelles données comportementales, que nous appelerons traces, sont collectées sur eux. En pratique, nous abandonnons plus nos traces que nous les fournissons de manière consentantes. Parce qu'elles semblent insignifiantes et innofensives, nous pouvons difficilement nous opposer frontalement à leur collection. Pourtant, en sachant combien de fois ais-je appelé un numéro de téléphone, mes derniers trajets, mes intéractions sur les réseaux sociaux, etc., il est possible de combiner, croiser et traiter ces traces pour en déduire des informations très intimes sur ma vie.

Les outils traditionnels, comme la cryptographie ou les VPNs, ont un impact limité contre la collecte de trace : le chiffrement ne cache que le contenu d'une conversation,

pas le fait qu'elle ai eu lieu à un moment donné entre deux individus. Pour empêcher un acteur malveillant de collecter ces traces, il est nécessaire d'anonymiser les personnes communicants sur le réseau. Décrit pour la première fois en 1981 par David Chaum, le réseau d'anonymat le plus utilisé aujourd'hui est Tor. Parmi les différentes architectures qui ont été proposées au fil des années, nous avons constaté que c'est celle qui proposait les meilleures performances pour les usagers qui s'est imposée. Ainsi, nous avançons que la qualité de service est un important facteur d'adoption pour les outils d'améliorations de la vie privée. Cependant malgré sa conception agnostique des protocoles, nous avons observé que Tor est souvent perçu comme limité à un usage web. Dans cette thèse, nous montrons à travers trois contributions qu'il est possible d'utiliser Tor pour d'autres usages tout en concervant la qualité de service qui a fait son succès.

État de l'art Les réseaux d'anonymats peuvent être classés en 3 grandes familles : *MixNets*, *DC-Net* et *Onion Routing*. Les deux premiers ont une propriété de sécurité supplémentaire, la résistance à un acteur malveillant particulier nommé attaquant global passif (nommé GPA). Le GPA peut écouter tout le réseau mais ne peut pas intéragir avec. Résister à un GPA est particulièrement complexe et couteux parce qu'il est nécessaire pour les participants d'envoyer du traffic factice ou de retarder les messages. En n'incluant pas la résistance aux GPA, l'*Onion Routing* est aussi plus facile à optimiser afin d'obtenir une qualité de service optimale, dont Tor est l'exemple le plus connu et explique ses bonnes performances.

Bien que la conception en *Onion Routing* de Tor présente moins de contraintes que les autres familles, sa mise en œuvre apporte son lot de défis qui ont été largement étudiées par la communauté scientifique. Nous les compilons en 12 axes d'études pour améliorer les performances du système Tor dans sa globalité, répartis en 5 thématiques : plus de relais, de meilleurs circuits, de meilleurs relais, un meilleur transport, et des applications mieux adaptées. Suitr à cette analyse, nous décidons de focaliser nos efforts dans le cadre de cette thèse sur l'amélioration du transport *via* l'utilisation du multi-chemin adapté aux contraintes spécifiques de Tor.

En parallèle, si Tor permet tout à chacun de communiquer directement en s'affranchissant des problèmes de pare-feu, de NAT et d'itinérance, il est limité à des communications deux à deux. Pour les communications de groupe, il est courant d'avoir recourt à un fournisseur de service pour centraliser les échanges, au risque de lui permettre de collecter des traces sur notre comportement. Pour ouvrir Tor a des communications de groupe sans compromis

sur la vie privée, nous choisissons plutôt des algorithmes de *rumor spreading* ne nécessitant pas d'infrastructure tiers pour fonctionner. Ces derniers nécessitent un compromis entre surconsommation réseau et délai d'acheminement qui limite traditionnellement leur adoption. Nous avons identifié l'utilisation de codage réseau comme prometeuse pour dépasser ce compromis mais nous constatons que les travaux théoriques ne sont pas appliquables du fait de leurs hypothèses de départ.

Donar est un client VoIP fonctionnant sur le réseau Tor existant qui satisfait aux standard de l'industrie sur la qualité des appels. Partant d'une analyse conduite sur le réseau Tor, nous avons observé que tous les circuits sur Tor présentaient une partie du temps des performances acceptable pour la VoIP mais aussi des pics de latence imprédictibles. Les approches taditionnelles basées sur la sélection de relais ou de circuits à la création ne sont donc pas fonctionnelles car elles se contentent de tester si un circuit subit un pic de latence lors de sa création. Avec Donar, nous proposons un système dynamique composé de plusieurs chemins : le trafic est partagé et entre plusieurs chemins, ce qui permet d'identifier les chemins les plus rapides. Ces derniers continuent d'être utilisés, là où les plus lents sont remplacés. En effet, Donar créer une réserve de circuits fixes importante au démarrage et pioche dedans pour tester de nouveaux circuits ; les circuits trop lents sont remis dans la réserve pour être retestés plus tard. En ayant la capacité de tester tout ces liens chaque seconde, Donar a une bien plus grande réactivité que les systèmes existants. En pratique, il est possible de passer des appels de 90 minutes sans aucun pic de latence dans plus de 90% des cas.

eTor est une solution qui permet d'étendre la bande passante totale du réseau Tor en tirant parti des relais hébergés en bordure du réseau. Cette augmentation de la bande passante permettrait d'envisager des nouveaux usages à travers Tor, comme le transfert de fichier ou la diffusion vidéo. Alors qu'un grand nombre de travaux s'est orienté autour de l'introduction de récompenses pour les personnes hébergeant des relais Tor, nous avançons qu'une frange ignorée des usagers de Tor hébergent déjà des relais chez eux. En effet, en explorant le consensus Tor, nous avons observé une part significative de relais résidentiels moins bien connectés. Aujourd'hui, ce type d'hébergement est découragé et Tor n'est pas capable d'en tirer partie de manière optimale. Avec eTor, nous proposons de passer à un système de circuit basique, dont la perte d'un relai est fatale, à un système de circuits entrelacés, qui peut supporter la perte d'un grand nombre de relais. Dans un premier

temps, nous montrons comment un tel système peut être intégré dans le système Tor. Dans un second temps, nous proposons une analyse de sécurité de cette nouvelle approche et montrons que dans un réseau avec un grand nombre de relais non-optimalement connectés, il est plus sécurisé de passer à des circuits entrelacés. Nous évaluons l'efficacité de eTor via un prototype et des simulations basées sur les données dans le consensus Tor pour illustrer à la fois les gains de performances et de sécurité.

CHEPIN est un protocole *gossip* optimisé grâce au codage réseau. Les algorithmes *gossip* permettent de réaliser des communications de groupe sans tiers. Dans le cadre de Tor, nous pensons que ce type d'algorithme est intéressant car la présence d'un tiers permet la collection de traces. Avec CHEPIN, nous proposons un algorithme gossip utilisant de codage réseau et fonctionnant dans le même environnement et avec les mêmes hypothèses de départ que les algorithmes *gossip* traditionnels. Nous avons observé expérimentalement que notre algorithme de codage réseau permettait d'améliorer à la fois les délais et de réduire la consommation réseau. Nous avons également montré que notre algorithme était moins sensible aux paramètres de l'algorithme, rendant son intégration possible dans un environnement où la taille du réseau et la fréquence de l'échange de message n'est pas connue à l'avance ou quand il est très variable.

Ces différents travaux nous ont permis d'entrevoir de nouveaux usages pour Tor tout en comprenant mieux ses faiblesses. Nous pensons que l'idée fondamentale de Donar d'observer et réagir en temps réel pourrait être encore d'avantage appronfie et étendue à travers une approche en boite blanche où l'on pourrait modéliser les différentes sources de latence mais aussi en rafinant les algorithmes d'ordonancement. Nous avons également observé des synergies entre Donar et eTor. Aujourd'hui, l'ordonanceur placé au sein des relais eTor est relativement naïf, se contenant d'envoyer les informations sur le lien qui semble le moins congestionné. En tenant compte de la diversité des trafic réseaux et des résultats obtenus par Donar, il est possible d'imaginer des ordonanceurs différents en fonction du trafic, observant et réagissant à des métriques différentes.

Bien que nous concevons CHEPIN comme la première pierre pour concevoir un algorithme de communication de groupe pour Tor, il reste encore e nombreux défis à relever. Premièrement, afin d'élargir son usage à des utilisateurs ne se faisant pas confiance, nous aimerions étudier les possibilités de le rendre résistant aux attaques Sybil, Eclipse et Byzantine. De plus, nous aimerions étudier des déploiements dans un environnement

non contrôlé sur Tor. À ce sujet, nous proposons plusieurs pistes à explorer pour rendre l'algorithme plus polyvalent.

En conclusion, cette thèse trace la voie pour rendre possible des usages sur Tor souvent pensés impossible comme les appels vocaux, les téléchargements, la diffusion de vidéo et les communication de groupe. Tout comme la navigation sur le web, ces usages génèrent eux aussi des traces silencieusement qui peuvent être utilisées au détriment des individus et de la société, et donc gagnent à profiter de l protection de Tor. Nous espérons que ces travaux participeront à redonner du contrôle aux internautes sur leur vie privée.

Bibliography

[1] Ikhlaq Rehman, « Facebook-Cambridge Analytica data harvesting: What you need to know », *in*: *Library Philosophy and Practice (e-journal)* (Jan. 2019), URL: https://digitalcommons.unl.edu/libphilprac/2497.

[2] David Sumpter, *Why the Facebook data available to Cambridge Analytica could not be used to target personalities in...* en, June 2018, URL: https://medium.com/@Soccermatics/why-the-facebook-data-available-to-cambridge-analytica-could-not-be-used-to-target-personalities-in-2904fa0571bd (visited on 08/19/2020).

[3] Wu Youyou, Michal Kosinski, and David Stillwell, « Computer-based personality judgments are more accurate than those made by humans », en, *in*: *Proceedings of the National Academy of Sciences* 112.*4* (Jan. 2015), Publisher: National Academy of Sciences Section: Social Sciences, pp. 1036–1040, ISSN: 0027-8424, 1091-6490, DOI: 10.1073/pnas.1418680112, URL: https://www.pnas.org/content/112/4/1036 (visited on 08/18/2020).

[4] Carole Cadwalladr and Emma Graham-Harrison, « Revealed: 50 million Facebook profiles harvested for Cambridge Analytica in major data breach », en-GB, *in*: *The Guardian* (Mar. 2018), ISSN: 0261-3077, URL: https://www.theguardian.com/news/2018/mar/17/cambridge-analytica-facebook-influence-us-election (visited on 08/19/2020).

[5] Matthew Rosenberg, Nicholas Confessore, and Carole Cadwalladr, « How Trump Consultants Exploited the Facebook Data of Millions », en-US, *in*: *The New York Times* (Mar. 2018), ISSN: 0362-4331, URL: https://www.nytimes.com/2018/03/17/us/politics/cambridge-analytica-trump-campaign.html (visited on 08/19/2020).

[6] Jan Martínez Ahrens, « La compañía que burló la intimidad de 50 millones de estadounidenses », es, *in*: *El País* (Mar. 2018), ISSN: 1134-6582, URL: https://elpais.com/internacional/2018/03/20/estados_unidos/1521574139_109464.html (visited on 08/19/2020).

[7] Patrick Beuth, « Was treibt eigentlich Cambridge Analytica? », de, *in*: (), URL: https://www.spiegel.de/netzwelt/netzpolitik/cambridge-analytica-das-steckt-hinter-der-datenanalyse-firma-a-1198962.html (visited on 08/19/2020).

[8] Brian Ries, Amanda Wills, and Veronica Rocha, *Live: Mark Zuckerberg testifies before Congress*, en, URL: https://www.cnn.com/politics/live-news/mark-zuckerberg-testifies-congress/index.html (visited on 08/27/2020).

[9] Chloe Watson, « The key moments from Mark Zuckerberg's testimony to Congress », en-GB, *in*: *The Guardian* (Apr. 2018), ISSN: 0261-3077, URL: https://www.theguardian.com/technology/2018/apr/11/mark-zuckerbergs-testimony-to-congress-the-key-moments (visited on 08/19/2020).

[10] Consumer Action, « Protect your phone records », en, *in*: (), p. 2.

[11] Jonathan Mayer, Patrick Mutchler, and John C. Mitchell, « Evaluating the privacy properties of telephone metadata », en, *in*: *Proceedings of the National Academy of Sciences* 113.*20* (May 2016), pp. 5536–5541, ISSN: 0027-8424, 1091-6490, DOI: 10.1073/pnas.1508081113, URL: http://www.pnas.org/lookup/doi/10.1073/pnas.1508081113 (visited on 05/27/2020).

[12] *Vodafone Australia admits hacking Fairfax journalist's phone*, en, Library Catalog: www.theguardian.com Section: Business, Sept. 2015, URL: http://www.theguardian.com/business/2015/sep/13/vodafone-australia-admits-hacking-fairfax-journalists-phone (visited on 05/29/2020).

[13] Ross Coulthart, *Metadata access is putting whistleblowers, journalists and democracy at risk*, en, May 2015, URL: https://www.smh.com.au/opinion/metadata-access-is-putting-whistleblowers-journalists-and-democracy-at-risk-20150504-1mzfi0.html (visited on 05/28/2020).

[14] David Kaplan, *Suspicions and Spies in Silicon Valley*, 2006, URL: https://www.newsweek.com/suspicions-and-spies-silicon-valley-109827.

[15] Ewen MacAskill et al., *NSA files decoded: Edward Snowden's surveillance revelations explained*, en, Section: US news, Nov. 2013, URL: http://www.theguardian.com/world/interactive/2013/nov/01/snowden-nsa-files-surveillance-revelations-decoded (visited on 08/19/2020).

[16] *Have I Been Pwned: Check if your email has been compromised in a data breach*, URL: https://haveibeenpwned.com/ (visited on 08/20/2020).

[17] Joseph Cox, *Leaked Documents Expose the Secretive Market for Your Web Browsing Data*, en, URL: https://www.vice.com/en_us/article/qjdkq7/avast-antivirus-sells-user-browsing-data-investigation (visited on 08/20/2020).

[18] Colin J. Bennett, « In Defense of Privacy: The Concept and the Regime », en-US, *in*: *Surveillance & Society* 8.*4* (Mar. 2011), pp. 485–496, ISSN: 1477-7487, DOI: 10.24908/ss.v8i4.4184, URL: https://ojs.library.queensu.ca/index.php/surveillance-and-society/article/view/4184 (visited on 08/21/2020).

[19] John Gilliom and Torin Monahan, *SuperVision: an introduction to the surveillance society*, Chicago: The University of Chicago Press, 2013, ISBN: 978-0-226-92443-4 978-0-226-92444-1.

[20] Daniel J. Solove, *'I've Got Nothing to Hide' and Other Misunderstandings of Privacy*, en, SSRN Scholarly Paper ID 998565, Rochester, NY: Social Science Research Network, July 2007, URL: https://papers.ssrn.com/abstract=998565 (visited on 08/19/2020).

[21] Shoshana Zuboff, *Big Other: Surveillance Capitalism and the Prospects of an Information Civilization*, en, SSRN Scholarly Paper ID 2594754, Rochester, NY: Social Science Research Network, Apr. 2015, URL: https://papers.ssrn.com/abstract=2594754 (visited on 08/25/2020).

[22] Antoinette Rouvroy and Thomas Berns, « Gouvernementalité algorithmique et perspectives d'émancipation », fr, *in*: *Reseaux* n° 177.*1* (May 2013), Publisher: La Découverte, pp. 163–196, ISSN: 0751-7971, URL: https://www.cairn.info/revue-reseaux-2013-1-page-163.htm (visited on 08/25/2020).

[23] Samuel D. Warren and Louis D. Brandeis, « The Right to Privacy », *in*: *Harvard Law Review* 4.*5* (1890), Publisher: The Harvard Law Review Association, pp. 193–220, ISSN: 0017-811X, DOI: 10.2307/1321160, URL: https://www.jstor.org/stable/1321160 (visited on 08/21/2020).

[24] *What is personal data?*, en, Text, URL: https://ec.europa.eu/info/law/law-topic/data-protection/reform/what-personal-data_en (visited on 08/28/2020).

[25] *Le Conseil d'État autorise la CNIL à ignorer le RGPD*, fr-FR, Section: Données personnelles, Oct. 2019, URL: https://www.laquadrature.net/2019/10/17/le-conseil-detat-autorise-la-cnil-a-ignorer-le-rgpd/ (visited on 08/28/2020).

[26] *Consentement : le pire de l'expérience utilisateur et de la surveillance avec Lemonde.fr*, fr, URL: https://www.pixeldetracking.com/fr/le-pire-du-recueil-du-consentement-avec-lemonde-fr (visited on 09/01/2020).

[27] Antoinette Rouvroy and Thomas Berns, « Le nouveau pouvoir statistique », fr, *in*: *Multitudes* n° 40.*1* (Feb. 2010), Publisher: Association Multitudes, pp. 88–103, ISSN: 0292-0107, URL: https://www.cairn.info/revue-multitudes-2010-1-page-88.html (visited on 08/28/2020).

[28] Erwan Le Merrer and Gilles Trédan, « Remote explainability faces the bouncer problem », en, *in*: *Nature Machine Intelligence* (Aug. 2020), Publisher: Nature Publishing Group, pp. 1–11, ISSN: 2522-5839, DOI: 10.1038/s42256-020-0216-z, URL: https://www.nature.com/articles/s42256-020-0216-z (visited on 09/01/2020).

[29] *Usage de Privacy-enhancing Technologies (PETs)*, fr, URL: https://cnpd.public.lu/fr/dossiers-thematiques/nouvelles-tech-communication/privacy-by-design/Usage-de-Privacy-enhancing-Technologies-_PETs_.html (visited on 09/01/2020).

[30] *Privacy enhancing technologies*, en-gb, Topic, URL: https://www.enisa.europa.eu/topics/data-protection/privacy-enhancing-technologies (visited on 09/01/2020).

[31] Office of the Privacy Commissioner of Canada, *Privacy Enhancing Technologies – A Review of Tools and Techniques*, eng, Last Modified: 2017-11-15, Nov. 2017, URL: https://www.priv.gc.ca/en/opc-actions-and-decisions/research/explore-privacy-research/2017/pet_201711/ (visited on 09/01/2020).

[32] Michael Backes et al., « AnoA: A framework for analyzing anonymous communication protocols », *in*: *2013 IEEE 26th Computer Security Foundations Symposium*, IEEE, 2013, pp. 163–178.

[33] *Block ISP tracking for good with IPVanish VPN*, URL: https://www.ipvanish.com/isp-tracking/ (visited on 07/06/2020).

[34] *How to block ISP tracking | NordVPN*, Library Catalog: nordvpn.com, June 22, 2020, URL: https://nordvpn.com/blog/isp-tracking/ (visited on 07/06/2020).

[35] Tim Tremblay, *How to Block ISP Tracking and Hide Internet Activity the Right Way*, Fastest VPN Guide, Library Catalog: www.fastestvpnguide.com, May 16, 2019, URL: https://www.fastestvpnguide.com/how-to-block-isp-tracking/ (visited on 07/06/2020).

[36] *Lantern - Open Internet for All*, URL: https://lantern.io/en_US/index.html (visited on 07/06/2020).

[37] *Psiphon | Uncensored Internet access for Windows and Mobile*, URL: https://psiphon3.com/en/index.html (visited on 07/06/2020).

[38] *Got Ethics A/S - Europe's fastest growing whistleblowing solutions provider*, Got Ethics A/S, Library Catalog: www.gotethics.com, URL: https://www.gotethics.com (visited on 07/06/2020).

[39] *Whispli – The Whistleblowing Platform for Security-Conscious Organizations*, Whispli, Library Catalog: whispli.com, URL: https://whispli.com/ (visited on 07/06/2020).

[40] *The whistleblowing system for those who care. WhistleB*, WhistleB, Library Catalog: whistleb.com, URL: https://whistleb.com/ (visited on 07/06/2020).

[41] Jon Porter, *Facebook pulls the plug on its data snooping Onavo VPN service*, en, Library Catalog: www.theverge.com, Feb. 2019, URL: https://www.theverge.com/2019/2/22/18235908/facebook-onavo-vpn-privacy-service-data-collection (visited on 06/12/2020).

[42] *NordVPN confirms it was hacked*, en-US, Library Catalog: techcrunch.com, URL: https://social.techcrunch.com/2019/10/21/nordvpn-confirms-it-was-hacked/ (visited on 06/12/2020).

[43] David L. Chaum, « Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms », *in*: *Commun. ACM* 24.*2* (Feb. 1981), pp. 84–90.

[44] David Chaum, « The dining cryptographers problem: Unconditional sender and recipient untraceability », *in*: *Journal of cryptology* (1988).

[45] Albert Kwon et al., « Riffle: An Efficient Communication System With Strong Anonymity », *in*: *PoPETs* (2016).

[46] David Lazar, Yossi Gilad, and Nickolai Zeldovich, « Karaoke: Distributed private messaging immune to passive traffic analysis », *in*: *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 2018.

[47] David Lazar, Yossi Gilad, and Nickolai Zeldovich, « Yodel: strong metadata security for voice calls », *in*: *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, 2019.

[48] Stevens Le Blond et al., « Towards efficient traffic-analysis resistant anonymity networks », *in*: *ACM SIGCOMM Computer Communication Review* (2013).

[49] Ania M Piotrowska et al., « The Loopix anonymity system », *in*: *26th USENIX Security Symposium*, 2017.

[50] Nirvan Tyagi et al., « Stadium: A Distributed Metadata-Private Messaging System », *in*: *26th Symposium on Operating Systems Principles*, SOSP, ACM, 2017.

[51] Jelle Van Den Hooff et al., « Vuvuzela: Scalable private messaging resistant to traffic analysis », *in*: *Proceedings of the 25th Symposium on Operating Systems Principles*, 2015.

[52] Philippe Golle and Ari Juels, « Dining cryptographers revisited », *in*: *International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, 2004.

[53] David Isaac Wolinsky et al., « Dissent in numbers: Making strong anonymity scale », *in*: *P10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*, 2012.

[54] Henry Corrigan-Gibbs and Bryan Ford, « Dissent: accountable anonymous group messaging », *in*: *Proceedings of the 17th ACM conference on Computer and communications security*, 2010.

[55] Henry Corrigan-Gibbs, Dan Boneh, and David Mazières, « Riposte: An anonymous messaging system handling millions of users », *in*: *2015 IEEE Symposium on Security and Privacy*, IEEE, 2015.

[56] Henry Corrigan-Gibbs, David Isaac Wolinsky, and Bryan Ford, « Proactively accountable anonymous messaging in Verdict », *in*: *Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13)*, 2013.

[57] Chen Chen et al., « HORNET: High-speed onion routing at the network layer », *in*: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015.

[58] Roger Dingledine, Nick Mathewson, and Paul Syverson, *Tor: The Second-Generation Onion Router*, tech. rep., Naval Research Lab Washington DC, 2004.

[59] Michael J. Freedman and Robert Morris, « Tarzan: A Peer-to-peer Anonymizing Network Layer », *in*: *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, 2002.

[60] David M Goldschlag, Michael G Reed, and Paul F Syverson, « Hiding routing information », *in*: *International workshop on information hiding*, Springer, 1996, pp. 137–150.

[61] Saikrishna Gumudavally et al., « HECTor: Homomorphic Encryption Enabled Onion Routing », *in*: *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, ICC 2019 - 2019 IEEE International Conference on Communications (ICC), ISSN: 1938-1883, May 2019, pp. 1–6, DOI: 10.1109/ICC.2019.8762038.

[62] Stevens Le Blond et al., « Herd: A scalable, traffic analysis resistant anonymity network for VoIP systems », *in*: *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM, 2015.

[63] Valerio Schiavoni, Etienne Rivière, and Pascal Felber, « Whisper: Middleware for confidential communication in large-scale networks », *in*: *31st International Conference on Distributed Computing Systems*, ICDCS, IEEE, 2011.

[64] Mike Perry, *Tor's Open Research Topics: 2018 Edition | Tor Blog*, 2018, URL: https://blog.torproject.org/tors-open-research-topics-2018-edition (visited on 07/08/2019).

[65] Roger Dingledine and Steven J Murdoch, « Performance Improvements on Tor or, Why Tor is slow and what we're going to do about it », *in*: *Online: http://www.torproject. org/press/presskit/2009-03-11-performance. pdf* (2009).

[66] *WeTransfer Case Study*, en-US, Library Catalog: aws.amazon.com, URL: https://aws.amazon.com/solutions/case-studies/wetransfer/ (visited on 06/04/2020).

[67] The Tor Project, *How can I share files anonymously through Tor?*, URL: https://2019.www.torproject.org/docs/faq.html.en#FileSharing (visited on 11/13/2020).

[68] Yérom-David Bromberg, Quentin Dufour, and Davide Frey, « Multisource Rumor Spreading with Network Coding », *in*: *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, IEEE, 2019, pp. 2359–2367.

[69] Pascal Felber, Anne-Marie Kermarrec, Lorenzo Leonini, et al., « Pulp: An adaptive gossip-based dissemination protocol for multi-source message streams », *in*: *Peer-to-Peer Networking and Applications* 5.*1* (2012), pp. 74–91.

[70] *The Anonymizer*, URL: http://anonymizer.com/.

[71] Oliver Berthold, Hannes Federrath, and Stefan Köpsell, « Web MIXes: A system for anonymous and unobservable Internet access », *in*: *Designing privacy enhancing technologies*, Springer, 2001, pp. 115–129.

[72] Web Dai, *Pipenet 1.1*, 1996, URL: http://www.weidai.com/pipenet.txt (visited on 09/15/2020).

[73] Andreas Pfitzmann, Birgit Pfitzmann, and Michael Waidner, « ISDN-Mixes: Untraceable Communication with Very Small Bandwidth Overhead », *in*: *Kommunikation in verteilten Systemen*, Springer, 1991, pp. 451–463.

[74] Michael G Reed, Paul F Syverson, and David M Goldschlag, « Anonymous connections and onion routing », *in*: *IEEE Journal on Selected areas in Communications* 16.*4* (1998), pp. 482–494.

[75] Paul F Syverson, Michael G Reed, and David M Goldschlag, « Onion Routing access configurations », *in*: *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*, vol. 1, IEEE, 2000, pp. 34–40.

[76] Paul Syverson et al., « Towards an analysis of onion routing security », *in*: *Designing Privacy Enhancing Technologies*, Springer, 2001, pp. 96–114.

[77] Marc Rennhard and Bernhard Plattner, « Introducing MorphMix: peer-to-peer based anonymous Internet usage with collusion detection », *in*: *Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society*, 2002, pp. 91–102.

[78] Michael K Reiter and Aviel D Rubin, « Crowds: Anonymity for web transactions », *in*: *ACM transactions on information and system security (TISSEC)* 1.*1* (1998), pp. 66–92.

[79] Zach Brown, « Cebolla: Pragmatic ip anonymity », *in*: *Ottawa Linux Symposium*, 2002, p. 55.

[80] Marc Rennhard et al., « An architecture for an anonymity network », *in*: *Proceedings Tenth IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises. WET ICE 2001*, IEEE, 2001, pp. 165–170.

[81] Prateek Mittal and Nikita Borisov, « Shadowwalker: peer-to-peer anonymous communication using redundant structured topologies », *in*: *Proceedings of the 16th ACM conference on Computer and communications security*, 2009, pp. 161–172.

[82] Paul Syverson, « A peel of onion », *in*: *Proceedings of the 27th Annual Computer Security Applications Conference*, 2011, pp. 123–137.

[83] George Danezis and Paul Syverson, « Bridging and fingerprinting: Epistemic attacks on route selection », *in*: *International Symposium on Privacy Enhancing Technologies Symposium*, Springer, 2008, pp. 151–166.

[84] Parisa Tabriz and Nikita Borisov, « Breaking the collusion detection mechanism of MorphMix », *in*: *International Workshop on Privacy Enhancing Technologies*, Springer, 2006, pp. 368–383.

[85] Max Schuchard et al., « Balancing the shadows », *in*: *Proceedings of the 9th annual ACM workshop on Privacy in the electronic society*, 2010, pp. 1–10.

[86] Mashael AlSabah and Ian Goldberg, « Performance and security improvements for tor: A survey », *in*: *ACM Computing Surveys (CSUR)* 49.*2* (2016), pp. 1–36.

[87] Mehran Alidoost Nia and Antonio Ruiz-Martinez, « Systematic literature review on the state of the art and future research work in anonymous communications systems », *in*: *Computers & electrical engineering* 69 (2018), pp. 497–520.

[88] Prateek Mittal et al., « PIR-Tor: Scalable Anonymous Communication Using Private Information Retrieval. », *in*: *USENIX Security Symposium*, 2011, pp. 31–31.

[89] Jon McLachlan et al., « Scalable onion routing with torsk », *in*: *Proceedings of the 16th ACM conference on Computer and communications security*, 2009, pp. 590–599.

[90] Chelsea Komlo, Nick Mathewson, and Ian Goldberg, « Walking Onions: Scaling Anonymity Networks while Protecting Users », *in*: (2020).

[91] Sajin Sasy and Ian Goldberg, « ConsenSGX: Scaling Anonymous Communications Networks with Trusted Execution Environments », en, *in*: *Proceedings on Privacy Enhancing Technologies* 2019.*3* (July 2019), pp. 331–349, ISSN: 2299-0984, DOI: 10.2478/popets-2019-0050.

[92] Garrett Hardin, « The Tragedy of the Commons », en, *in*: *Science* 162.*3859* (Dec. 1968), pp. 1243–1248, ISSN: 0036-8075, 1095-9203, DOI: 10.1126/science.162.3859.1243, URL: https://www.sciencemag.org/lookup/doi/10.1126/science.162.3859.1243 (visited on 09/24/2020).

[93] Rob Jansen, Paul Syverson, and Nicholas Hopper, « Throttling Tor bandwidth parasites », *in*: *Presented as part of the 21st {USENIX} Security Symposium ({USENIX} Security 12)*, 2012, pp. 349–363.

[94] W Brad Moore, Chris Wacek, and Micah Sherr, « Exploring the potential benefits of expanded rate limiting in tor: Slow and steady wins the race with tortoise », *in*: *Proceedings of the 27th Annual Computer Security Applications Conference*, 2011, pp. 207–216.

[95] Roger Dingledine, Dan S Wallach, et al., « Building incentives into Tor », *in*: *International Conference on Financial Cryptography and Data Security*, Springer, 2010, pp. 238–256.

[96] Rob Jansen, Nicholas Hopper, and Yongdae Kim, « Recruiting new Tor relays with BRAIDS », *in*: *Proceedings of the 17th ACM conference on Computer and communications security*, 2010, pp. 319–328.

[97] Rob Jansen, Aaron Johnson, and Paul Syverson, *LIRA: Lightweight incentivized routing for anonymity*, tech. rep., NAVAL RESEARCH LAB WASHINGTON DC, 2013.

[98] Rob Jansen et al., *From onions to shallots: Rewarding Tor relays with TEARS*, tech. rep., NAVAL RESEARCH LAB WASHINGTON DC, 2014.

[99] Dario Catalano, Dario Fiore, and Rosario Gennaro, « Certificateless onion routing », *in*: *Proceedings of the 16th ACM conference on Computer and communications security*, 2009, pp. 151–160.

[100] Aniket Kate, Greg Zaverucha, and Ian Goldberg, « Pairing-based onion routing », *in*: *International Workshop on Privacy Enhancing Technologies*, Springer, 2007, pp. 95–112.

[101] Lasse Øverlier and Paul Syverson, « Improving efficiency and simplicity of Tor circuit establishment and hidden services », *in*: *International Workshop on Privacy Enhancing Technologies*, Springer, 2007, pp. 134–152.

[102] Ian Goldberg, Douglas Stebila, and Berkant Ustaoglu, « Anonymity and one-way authentication in key exchange protocols », *in*: *Designs, Codes and Cryptography* 67.*2* (2013), pp. 245–269.

[103] Michael Backes, Aniket Kate, and Esfandiar Mohammadi, « Ace: an efficient key-exchange protocol for onion routing », *in*: *Proceedings of the 2012 ACM workshop on Privacy in the electronic society*, 2012, pp. 55–64.

[104] Frank Cangialosi, Dave Levin, and Neil Spring, « Ting: Measuring and exploiting latencies between all tor nodes », *in*: *Proceedings of the 2015 Internet Measurement Conference*, 2015, pp. 289–302.

[105] Micah Sherr, Matt Blaze, and Boon Thau Loo, « Scalable link-based relay selection for anonymous routing », *in*: *International Symposium on Privacy Enhancing Technologies Symposium*, Springer, 2009, pp. 73–93.

[106] Masoud Akhoondi, Curtis Yu, and Harsha V Madhyastha, « LASTor: A low-latency AS-aware Tor client », *in*: *2012 IEEE Symposium on Security and Privacy*, IEEE, 2012, pp. 476–490.

[107] Mohsen Imani, Mehrdad Amirabadi, and Matthew Wright, « Modified relay selection and circuit selection for faster Tor », *in*: *IET Communications* 13.*17* (2019), pp. 2723–2734.

[108] Prithula Dhungel et al., « Waiting for anonymity: Understanding delays in the Tor overlay », *in*: *2010 IEEE Tenth International Conference on Peer-to-Peer Computing (P2P)*, IEEE, 2010, pp. 1–4.

[109] Robin Snader and Nikita Borisov, « A Tune-up for Tor: Improving Security and Performance in the Tor Network. », *in*: *ndss*, vol. 8, 2008, p. 127.

[110] Tao Wang et al., « Congestion-aware path selection for Tor », *in*: *International Conference on Financial Cryptography and Data Security*, Springer, 2012.

[111] Armon Barton et al., « Towards predicting efficient and anonymous Tor circuits », *in*: *27th USENIX Security Symposium*, 2018.

[112] John Geddes, Mike Schliep, and Nicholas Hopper, « Abra cadabra: Magically increasing network utilization in tor by avoiding bottlenecks », *in*: *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society*, 2016, pp. 165–176.

[113] Robert Annessi and Martin Schmiedecker, « Navigator: Finding faster paths to anonymity », *in*: *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, IEEE, 2016, pp. 214–226.

[114] Mashael AlSabah et al., « DefenestraTor: Throwing out windows in Tor », *in*: *International Symposium on Privacy Enhancing Technologies Symposium*, Springer, 2011, pp. 134–154.

[115] HT Kung, Trevor Blackwell, and Alan Chapman, « Credit-based flow control for ATM networks: credit update protocol, adaptive credit allocation and statistical multiplexing », *in*: *Proceedings of the conference on Communications architectures, protocols and applications*, 1994, pp. 101–114.

[116] *#4086 (Compare performance of TokenBucketRefillInterval params in simulated network) – Tor Bug Tracker & Wiki*, URL: `https://trac.torproject.org/projects/tor/ticket/4086` (visited on 11/02/2020).

[117] K Kiran et al., « Optimal Token Bucket Refilling for Tor network », *in*: *2018 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, IEEE, 2018, pp. 1–6.

[118] Ellen L. Hahne, « Round-robin scheduling for max-min fairness in data networks », *in*: *IEEE Journal on Selected Areas in communications* 9.7 (1991), pp. 1024–1039.

[119] Florian Tschorsch and Björn Scheuermann, « Tor is unfair—And what to do about it », *in*: *2011 IEEE 36th Conference on Local Computer Networks*, IEEE, 2011, pp. 432–440.

[120] Can Tang and Ian Goldberg, « An improved algorithm for Tor circuit scheduling », *in*: *Proceedings of the 17th ACM conference on Computer and communications security*, 2010, pp. 329–339.

[121] Mashael AlSabah, Kevin Bauer, and Ian Goldberg, « Enhancing Tor's performance using real-time traffic classification », *in*: *Proceedings of the 2012 ACM conference on Computer and communications security*, 2012, pp. 73–84.

[122] Rob Jansen et al., « Never Been KIST: Tor's Congestion Management Blossoms with Kernel-Informed Socket Transport », *in*: *23rd USENIX Security Symposium*, 2014.

[123] Rob Jansen et al., « KIST: Kernel-Informed Socket Transport for Tor », *in*: *ACM Transactions on Privacy and Security (TOPS)* 22.*1* (2018), pp. 1–37.

[124] Steven J Murdoch, « Comparison of Tor datagram designs », *in*: *Technical report* (2011).

[125] Deepika Gopal and Nadia Heninger, « Torchestra: Reducing interactive traffic delays over Tor », *in*: *Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society*, 2012, pp. 31–42.

[126] Joel Reardon and Ian Goldberg, « Improving Tor using a TCP-over-DTLS Tunnel. », *in*: *USENIX Security Symposium*, 2009, pp. 119–134.

[127] Mashael AlSabah and Ian Goldberg, « PCTCP: per-circuit TCP-over-IPsec transport for anonymous communication overlay networks », *in*: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 349–360.

[128] John Geddes, Rob Jansen, and Nicholas Hopper, « IMUX: Managing tor connections from two to infinity, and beyond », *in*: *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, 2014, pp. 181–190.

[129] Camilo Viecco, « UDP-OR: A fair onion transport design », *in*: *Proceedings of Hot Topics in Privacy Enhancing Technologies (HOTPETS'08)* (2008).

[130] Marc Liberatore, *100-tor-spec-udp.txt proposals - torspec - Tor's protocol specifications*, URL: `https://gitweb.torproject.org/torspec.git/tree/proposals/100-tor-spec-udp.txt` (visited on 09/30/2020).

[131] Michael F Nowlan, David Isaac Wolinsky, and Bryan Ford, « Reducing latency in Tor circuits with unordered delivery », *in*: *3rd {USENIX} Workshop on Free and Open Communications on the Internet ({FOCI} 13)*, 2013.

[132] Mike Perry, *[tor-dev] The case for Tor-over-QUIC*, Mar. 2018, URL: `https://lists.torproject.org/pipermail/tor-dev/2018-March/013026.html` (visited on 09/30/2020).

[133] Mark Handley et al., *TCP Extensions for Multipath Operation with Multiple Addresses*, URL: `https://tools.ietf.org/html/rfc6824` (visited on 10/05/2020).

[134] Alexander Frommgen et al., « ReMP TCP: Low latency multipath TCP », *in*: *2016 IEEE International Conference on Communications (ICC)*, IEEE, 2016, pp. 1–7.

[135] Alexander Froemmgen, Jens Heuschkel, and Boris Koldehofe, « Multipath tcp scheduling for thin streams: Active probing and one-way delay-awareness », *in*: *2018 IEEE International Conference on Communications (ICC)*, IEEE, 2018, pp. 1–7.

[136] Olaf Landsiedel et al., « Dynamic multipath onion routing in anonymous peer-to-peer overlay networks », *in*: *IEEE GLOBECOM 2007-IEEE Global Telecommunications Conference*, IEEE, 2007, pp. 64–69.

[137] Hasan T Karaoglu et al., « Multi path considerations for anonymized routing: Challenges and opportunities », *in*: *2012 5th International Conference on New Technologies, Mobility and Security (NTMS)*, IEEE, 2012, pp. 1–5.

[138] Mashael AlSabah et al., « The path less travelled: Overcoming Tor's bottlenecks with traffic splitting », *in*: *International Symposium on Privacy Enhancing Technologies*, PETS, Springer, 2013.

[139] Lei Yang and Fengjun Li, « mtor: A multipath tor routing beyond bandwidth throttling », *in*: *2015 IEEE Conference on Communications and Network Security (CNS)*, IEEE, 2015, pp. 479–487.

[140] Wladimir De la Cadena et al., « Analysis of Multi-path Onion Routing-Based Anonymization Networks », *in*: *IFIP Annual Conference on Data and Applications Security and Privacy*, Springer, 2019, pp. 240–258.

[141] The Tor Project Inc, *Tor Project: FAQ*, URL: `https://2019.www.torproject.org/docs/faq.html.en#TransportIPnotTCP` (visited on 11/04/2020).

[142] ITU, *ITU-T Recommendation G.114, "One way transmission time"*, 2003, URL: `https://www.itu.int/rec/T-REC-G.114`.

[143] Alexandros Fakis, Georgios Karopoulos, and Georgios Kambourakis, « OnionSIP: Preserving Privacy in SIP with Onion Routing. », *in*: *J. UCS* (2017).

[144] Georgios Karopoulos, Alexandros Fakis, and Georgios Kambourakis, « Complete SIP message obfuscation: PrivaSIP over Tor », *in*: *2014 Ninth International Conference on Availability, Reliability and Security*, IEEE, 2014.

[145] Van Gegel, *TORFone: secure VoIP tool*, 2013, URL: `http://torfone.org/`.

[146] Pere Manils et al., « Compromising Tor anonymity exploiting P2P information leakage », *in*: *arXiv preprint arXiv:1004.1461* (2010).

[147] *OnionShare*, en, URL: `https://onionshare.org` (visited on 08/01/2020).

[148] Alan J. Demers, Daniel H. Greene, Carl Hauser, et al., « Epidemic Algorithms for Replicated Database Maintenance », *in*: *Operating Systems Review* 22.*1* (1988), pp. 8–32.

[149] Boris Koldehofe, « Simple gossiping with balls and bins. », *in*: *Stud. Inform. Univ.* 3.*1* (2004), pp. 43–60.

[150] Patrick Euster et al., « From epidemics to distributed computing », *in*: *IEEE Computer* 37.*5* (2004), pp. 60–67.

[151] Vinay Pai et al., « Chainsaw: Eliminating trees from overlay multicast », *in*: *International Workshop on Peer-to-Peer Systems*, Springer, 2005, pp. 127–140.

[152] Bo Li et al., « Inside the new coolstreaming: Principles, measurements and performance implications », *in*: *IEEE INFOCOM 2008-The 27th Conference on Computer Communications*, IEEE, 2008, pp. 1031–1039.

[153] Sujay Sanghavi, Bruce E. Hajek, and Laurent Massoulié, « Gossiping With Multiple Messages », *in*: *IEEE Trans. Information Theory* 53.*12* (2007), pp. 4640–4654.

[154] Mary-Luc Champel, Anne-Marie Kermarrec, and Nicolas Le Scouarnec, « FoG: Fighting the Achilles' Heel of Gossip Protocols with Fountain Codes », *in*: *SSS 2009, Lyon, France, November 3-6, 2009. Proceedings*, 2009, pp. 180–194.

[155] Philip A. Chou, Yunnan Wu, and Kamal Jain, « Practical Network Coding », *in*: *Allerton Conference on Communication, Control, and Computing*, Oct. 2003.

[156] Supratim Deb, Muriel Médard, and Clifford Choute, « Algebraic gossip: A network coding approach to optimal multiple rumor mongering », *in*: *IEEE/ACM Transactions on Networking (TON)* 14.*SI* (2006), pp. 2486–2507.

[157] Bernhard Haeupler, « Analyzing network coding gossip made easy », *in*: *Proceedings of the forty-third annual ACM symposium on Theory of computing*, ACM, 2011, pp. 293–302.

[158] C. Fragouli, J. Widmer, and J. Y. Le Boudec, « Efficient Broadcasting Using Network Coding », *in*: *IEEE/ACM Transactions on Networking* 16.*2* (Apr. 2008), pp. 450–463, ISSN: 1063-6692.

[159] Sachin Katti, Hariharan Rahul, Wenjun Hu, et al., « XORs in the air: Practical wireless network coding », *in*: *ACM SIGCOMM computer communication review*, vol. 36, ACM, 2006, pp. 243–254.

[160] Jan Camenisch and Anna Lysyanskaya, « A formal treatment of onion routing », *in*: *Annual International Cryptology Conference*, Springer, 2005, pp. 169–187.

[161] Andrei Serjantov and Peter Sewell, « Passive Attack Analysis for Connection-Based Anonymity Systems », en, *in*: *Computer Security – ESORICS 2003*, ed. by Einar Snekkenes and Dieter Gollmann, Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2003, pp. 116–131, ISBN: 978-3-540-39650-5.

[162] Aaron Johnson et al., « Users Get Routed: Traffic Correlation on Tor by Realistic Adversaries », en, *in*: *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security - CCS '13*, Berlin, Germany: ACM Press, 2013, pp. 337–348, ISBN: 978-1-4503-2477-9, DOI: 10.1145/2508859.2516651.

[163] Florentin Rochet and Olivier Pereira, « Waterfilling: Balancing the Tor Network with Maximum Diversity », en, *in*: *Proceedings on Privacy Enhancing Technologies* 2017.*2* (Apr. 2017), pp. 4–22, ISSN: 2299-0984, DOI: 10.1515/popets-2017-0013.

[164] David Isaac Wolinsky, Ewa Syta, and Bryan Ford, « Hang with your buddies to resist intersection attacks », *in*: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 1153–1166.

[165] Matthew K Wright et al., « The predecessor attack: An analysis of a threat to anonymous communications systems », *in*: *ACM Transactions on Information and System Security (TISSEC)* (2004).

[166] Matthew Wright et al., « Defending Anonymous Communications Against Passive Logging Attacks », *in*: *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, SP '03, USA: IEEE Computer Society, 2003, p. 28, ISBN: 0769519407.

[167] Gerry Wan et al., « Guard Placement Attacks on Path Selection Algorithms for Tor », *in*: *Proceedings on Privacy Enhancing Technologies* (2019).

[168] Prateek Mittal et al., « Stealthy traffic analysis of low-latency anonymous communication using throughput fingerprinting », *in*: *Proceedings of the 18th ACM conference on Computer and communications security*, 2011.

[169] Xiang Cai et al., « Touching from a distance: Website fingerprinting attacks and defenses », *in*: *Proceedings of the 2012 ACM conference on Computer and communications security*, 2012.

[170] Tobias Pulls and Rasmus Dahlberg, « Website Fingerprinting with Website Oracles », *in*: *Proceedings on Privacy Enhancing Technologies* (2020).

[171] Albert Kwon et al., « Circuit fingerprinting attacks: Passive deanonymization of tor hidden services », *in*: *24th USENIX Security Symposium (USENIX Security 15)*, 2015.

[172] Andriy Panchenko et al., « Analysis of fingerprinting techniques for tor hidden services », *in*: *Proceedings of the 2017 on Workshop on Privacy in the Electronic Society*, 2017.

[173] Alex Biryukov, Ivan Pustogarov, and Ralf-Philipp Weinmann, « Trawling for tor hidden services: Detection, measurement, deanonymization », *in*: *2013 IEEE Symposium on Security and Privacy*, IEEE, 2013.

[174] Nick Mathewson et al., *Tor Rendezvous Specification - Version 3*, 2017, URL: `https://gitweb.torproject.org/torspec.git/tree/rend-spec-v3.txt`.

[175] George Danezis and Richard Clayton, « Route fingerprinting in anonymous communications », *in*: *Sixth IEEE International Conference on Peer-to-Peer Computing (P2P'06)*, IEEE, 2006, pp. 69–72.

[176] Steven J Murdoch et al., *Tor: The Second-Generation Onion Router (2013 DRAFT v1)*, 2014, URL: `https://gitweb.torproject.org/tor-design-2012.git/`.

[177] Zhen Ling et al., « Equal-sized cells mean equal-sized packets in Tor? », *in*: *2011 IEEE International Conference on Communications (ICC)*, IEEE, 2011.

[178] GStreamer, *GStreamer: open source multimedia framework*, 2020, URL: `https://gstreamer.freedesktop.org/` (visited on 01/31/2020).

[179] Opus Codec, *Codec Landscape*, 2020, URL: `https://opus-codec.org/comparison/` (visited on 01/31/2020).

[180] ITU, *E.800 : Definitions of terms related to quality of service*, 2008, URL: https://www.itu.int/rec/T-REC-E.800-200809-I.

[181] ITU, *G.1028: End-to-end quality of service for voice over 4G mobile networks*, 2019, URL: https://www.itu.int/rec/T-REC-G.1028.

[182] J. Rosenberg et al., *SIP: Session Initiation Protocol*, Request for Comments (RFC) 3261, Internet Engineering Task Force (IETF), June 2002.

[183] H. Schulzrinne et al., *RTP: A Transport Protocol for Real-Time Applications*, Request for Comments (RFC) 3550, Internet Engineering Task Force (IETF), July 2003.

[184] JM. Valin, K. Vos, and T. Terriberry, *Definition of the Opus Audio Codec*, Request for Comments (RFC) 6716, Internet Engineering Task Force (IETF), Sept. 2012.

[185] JM. Valin and K. Vos, *Updates to the Opus Audio Codec*, RFC 8251, Oct. 2017.

[186] Christian Hoene et al., *Summary of Opus listening test results*, 2013, URL: https://tools.ietf.org/html/draft-ietf-codec-results-03 (visited on 01/31/2020).

[187] kamedo2, *Results of the public multiformat listening test*, 2014, URL: https://listening-test.coresv.net/results.htm (visited on 01/31/2020).

[188] Soren Vang Andersen et al., *Internet Low Bit Rate Codec (iLBC)*, Request for Comments (RFC) 3951, Internet Engineering Task Force (IETF), Dec. 2004.

[189] Luca Barbato, *RTP Payload Format for Vorbis Encoded Audio*, Request for Comments (RFC) 5215, Internet Engineering Task Force (IETF), Aug. 2008.

[190] G. Herlein et al., *RTP Payload Format for the Speex Codec*, Request for Comments (RFC) 5574, Internet Engineering Task Force (IETF), June 2009.

[191] Monty Icenogle, *T-mobile does have a hard 4 hour single call duration limit*, 2015, URL: https://kd6cae.livejournal.com/271120.html.

[192] Voyced, *Is there a maximum call length or duration*, 2019, URL: https://www.voyced.eu/clients/index.php/knowledgebase/397/Is-there-a-maximum-Call-length-or-duration.html (visited on 11/28/2019).

[193] Tim Terriberry and Koen Vos, *Definition of the Opus Audio Codec*, 2012, URL: https://tools.ietf.org/html/rfc6716#section-2.1.6 (visited on 01/31/2020).

[194] Yi Han et al., « Determination of bit-rate adaptation thresholds for the opus codec for VoIP services », *in*: *2014 IEEE Symposium on Computers and Communications (ISCC)*, IEEE, 2014.

[195] Katrin Schoenenberg et al., « On interaction behaviour in telephone conversations under transmission delay », *in*: *Speech Communication* (2014).

[196] Sue B Moon, Jim Kurose, and Don Towsley, « Packet audio playout delay adjustment: performance bounds and algorithms », *in*: *Multimedia systems* (1998).

[197] Byeong Hoon Kim et al., « VoIP receiver-based adaptive playout scheduling and packet loss concealment technique », *in*: *IEEE Transactions on consumer Electronics* (2013).

[198] Yi J Liang, Nikolaus Farber, and Bernd Girod, « Adaptive playout scheduling and loss concealment for voice communication over IP networks », *in*: *IEEE Transactions on Multimedia* (2003).

[199] Roger Dingledine, Nick Mathewson, and Paul Syverson, *Tor: The second-generation onion router*, tech. rep., Naval Research Lab Washington DC, 2004.

[200] Mike Perry, *The move to two guard nodes*, 2018, URL: `https://gitweb.torproject.org/user/mikeperry/torspec.git/tree/proposals/xxx-two-guard-nodes.txt?h=twoguards` (visited on 02/05/2020).

[201] David L Chaum, « Untraceable electronic mail, return addresses, and digital pseudonyms », *in*: *Communications of the ACM* (1981).

[202] Maimun Rizal, « A Study of VoIP performance in anonymous network-The onion routing (Tor) », PhD thesis, Niedersächsische Staats-und Universitätsbibliothek Göttingen, 2014.

[203] Stephan Heuser et al., « Phonion: Practical protection of metadata in telephony networks », *in*: *Proceedings on Privacy Enhancing Technologies* (2017).

[204] Piyush Kumar Sharma et al., « The road not taken: re-thinking the feasibility of voice calling over tor », *in*: *arXiv preprint arXiv:2007.04673* (2020).

[205] Nick Montfort et al., *Tor project feature tracker: Closed enhancement, "UDP over Tor"*, 2013, URL: `https://trac.torproject.org/projects/tor/ticket/7830`.

[206] Aaron Johnson et al., « Users get routed: Traffic correlation on Tor by realistic adversaries », *in*: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013.

[207] Steven J Murdoch and George Danezis, « Low-cost traffic analysis of Tor », *in*: *2005 IEEE Symposium on Security and Privacy (S&P'05)*, IEEE, 2005.

[208] Kevin Bauer et al., « On the optimal path length for Tor », *in*: *HotPets in conjunction with Tenth International Symposium on Privacy Enhancing Technologies (PETS 2010), Berlin, Germany*, 2010.

[209] *Proxy & VPN detection API - IPHub.info*, URL: `https://iphub.info/` (visited on 06/30/2020).

[210] Robin Snader and Nikita Borisov, « A Tune-up for Tor: Improving Security and Performance in the Tor Network », *in*: *Proceedings of the Network and Distributed System Security Symposium, NDSS 2008, San Diego, California, USA, 10th February - 13th February 2008*, 2008.

[211] Chris Wacek et al., « An Empirical Evaluation of Relay Selection in Tor », *in*: *20th Annual Network and Distributed System Security Symposium, NDSS 2013, San Diego, California, USA, February 24-27, 2013*, 2013.

[212] Tao Wang et al., « Congestion-Aware Path Selection for Tor », *in*: *Financial Cryptography and Data Security - 16th International Conference, FC 2012, Kralendijk, Bonaire, Februray 27-March 2, 2012, Revised Selected Papers*, 2012, pp. 98–113.

[213] Tariq Elahi et al., « Changing of the guards: a framework for understanding and improving entry guard selection in tor », *in*: *Proceedings of the 11th annual ACM Workshop on Privacy in the Electronic Society, WPES 2012, Raleigh, NC, USA, October 15, 2012*, ed. by Ting Yu and Nikita Borisov, ACM, 2012.

[214] J. A. Lockitt, A. G. Gatfield, and T. R. Dobyns, « A Selective Repeat ARQ System », *in*: *3rd International Conference on Digital Satellite Communications*, 1975, pp. 189–195.

[215] E. Weldon, « An Improved Selective-Repeat ARQ Strategy », *in*: *IEEE Transactions on Communications* 30.*3* (Mar. 1982), pp. 480–486, ISSN: 0090-6778, DOI: `10.1109/TCOM.1982.1095497`.

[216] Larry L. Peterson and Bruce S. Davie, *Computer Networks: A Systems Approach, 3rd Edition*, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003, pp. 97–110, ISBN: 978-1-55860-832-0.

[217] Márk Jelasity and Özalp Babaoglu, « T-Man: Gossip-Based Overlay Topology Management », *in*: *ESOA 2005, Utrecht, The Netherlands, July 25, 2005, Revised Selected Papers*, 2005, pp. 1–15.

[218] Simon Bouget et al., « Pleiades: Distributed Structural Invariants at Scale », *in*: *DSN 2018*, Luxembourg, Luxembourg: IEEE, June 2018, pp. 1–12.

[219] Avinash Lakshman and Prashant Malik, « Cassandra: A Decentralized Structured Storage System », *in*: *SIGOPS Oper. Syst. Rev.* 44.*2* (Apr. 2010), pp. 35–40, ISSN: 0163-5980.

[220] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, et al., « Dynamo: Amazon's Highly Available Key-value Store », *in*: *Proceedings of Twenty-first ACM SIGOPS*, SOSP '07, Stevenson, Washington, USA: ACM, 2007, pp. 205–220, ISBN: 978-1-59593-591-5.

[221] Armon Dadgar, James Phillips, and Jon Currey, « Lifeguard : SWIM-ing with Situational Awareness », *in*: *CoRR* abs/1707.00788 (2017).

[222] Abhinandan Das et al., « SWIM: Scalable Weakly-consistent Infection-style Process Group Membership Protocol », *in*: *In Proc. 2002 Intnl. Conf. DSN*, 2002, pp. 303–312.

[223] Márk Jelasity, Alberto Montresor, and Ozalp Babaoglu, « Gossip-based Aggregation in Large Dynamic Networks », *in*: *ACM Trans. Comput. Syst.* 23.*3* (Aug. 2005), pp. 219–252, ISSN: 0734-2071.

[224] Davide Frey et al., *Live Streaming with Gossip*, Research Report RR-9039, Inria Rennes Bretagne Atlantique ; RR-9039, Mar. 2017.

[225] Davide Frey et al., « Stretching gossip with live streaming », *in*: *DSN 2009, Estoril, Lisbon, Portugal, June 29 - July 2, 2009*, 2009, pp. 259–264.

[226] Elli Androulaki, Artem Barger, Vita Bortnikov, et al., « Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains », *in*: *CoRR* abs/1801.10228 (2018).

[227] Brice Nédelec, Pascal Molli, and Achour Mostefaoui, « CRATE: Writing Stories Together with Our Browsers », *in*: *Proceedings of the 25th International Conference Companion on World Wide Web*, 2016, pp. 231–234, ISBN: 978-1-4503-4144-8.

[228] Armando Castañeda, Sergio Rajsbaum, and Michel Raynal, « The renaming problem in shared memory systems: An introduction », *in*: *Computer Science Review* 5.*3* (2011), pp. 229–251.

[229] Christina Fragouli, Jean-Yves Le Boudec, and Jörg Widmer, « Network Coding: An Instant Primer », *in*: *SIGCOMM Comput. Commun. Rev.* 36.*1* (Jan. 2006), pp. 63–68, ISSN: 0146-4833.

[230] Márk Jelasity et al., « Gossip-based Peer Sampling », *in*: *TOCS* 25.*3* (2007).

[231] Leslie Lamport, « Time, clocks, and the ordering of events in a distributed system », *in*: *Communications of the ACM* 21.*7* (1978), pp. 558–565.

[232] Rui Zhu, Bang Liu, Di Niu, et al., « Network Latency Estimation for Personal Devices: A Matrix Completion Approach », *in*: *IEEE/ACM Trans. Netw.* 25.*2* (2017), pp. 724–737.

[233] Ranjita Bhagwan, Stefan Savage, and Geoffrey M. Voelker, « Understanding Availability », *in*: *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, 2003.

[234] John R Douceur, « The sybil attack », *in*: *International workshop on peer-to-peer systems*, Springer, 2002, pp. 251–260.

[235] Atul Singh et al., « Eclipse attacks on overlay networks: Threats and defenses », *in*: *In IEEE INFOCOM*, Citeseer, 2006.

[236] Edward Bortnikov et al., « Brahms: Byzantine resilient random membership sampling », *in*: *Computer Networks* 53.*13* (2009), pp. 2340–2359.

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

Titre : Réseaux en oignon haut débit et temps réel pour protéger la vie privée de tou·tes

Mot clés : vie privée, routage en oignon, multi-chemin, rumeurs, codage réseau

Résumé : De l'ingérence électorale à la manipulation, les problèmes de vie privée, ou surveillance, sont de plus en plus importants dans le débat public. La surveillance est rendue possible grâce à la collecte illimitée de données sur les comportements humains, souvent appellées traces. Nous explorons comment les réseaux d'anonymats peuvent protéger de la surveillance en empêchant la collection de traces. Nous avons observé que le réseau le plus utilisé, Tor, est aussi celui avec les meilleures performances. Tor souffre tout de même de limitations et est souvent limité à la navigation web anonyme. Pour élargir les usages de Tor, de la VoIP aux transferts de fichier en passant par les communication de groupe, nous avons exploité deux concepts : le multi-chemin et les rumeurs. Donar est un client VoIP fonctionnant sur le réseau Tor existant qui satisfait aux standard de l'industrie sur la qualité des appels grâce à un algorithme temps réel multi-chemin. eTor est une solution de transfert de fichiers qui permet de contribuer au réseau avec des équipements à domicile grâce à des mécanismes de tolérance aux pannes pour fournir la bande passante requise. CHEPIN libère les communications de groupe des serveurs grâce à un protocole gossip optimisé avec du codage réseau. Avec nos contributions nous avons pour objectif de tracer la voie pour démocratiser les réseaux d'anonymats et ainsi aider les gens à se protéger de la surveillance.

Title: High-throughput real-time onion networks to protect everyone's privacy

Keywords: privacy, onion routing, multipath, gossip, network coding

Abstract: From electoral interference to manipulation, privacy issues, or surveillance, gain more and more traction in the public debate. Surveillance is possible thanks to unlimited data collection on human behaviors, often referred as traces. We explore how anonymity networks could extend people protection against surveillance by preventing traces collection. We observed that the most used network, Tor, is also the one that features the best performances. Tor still suffers from limitations and often restrained to anonymous web browsing. To widen Tor usage, from VoIP to file transfer including group communication, we leveraged two main concepts: multipath and gossip. Donar is a VoIP client running over legacy Tor meeting industry standards for calls quality thanks to a real-time multipath algorithm. eTor is a file transfer solution that enable contribution to the network with home devices thanks to a fault tolerance mechanisms to provide the required bandwidth. CHEPIN frees group communication from servers thanks to a gossip protocol optimized with network coding. With our contributions, we aim to pave the way to democratizing anonymity networks and help people protect themself against surveillance.