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 Piaud 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

Chapter 1

Introduction

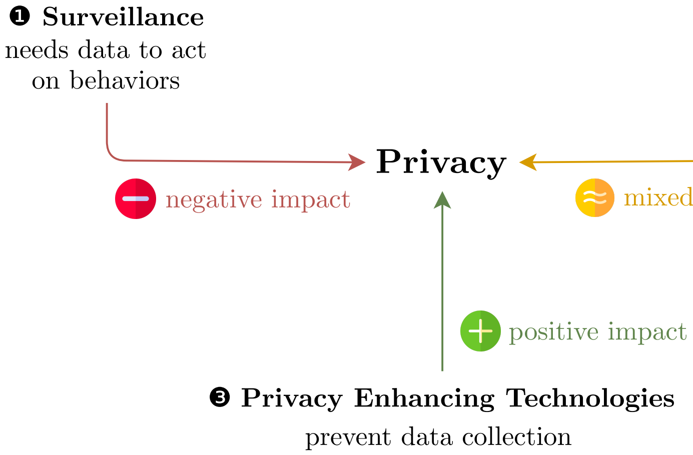


Figure 1.1 – Influence of surveillance, law and privacy enhancing technologies on privacy.

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 [178]. While the real impact of this scandal on people's behavior is still not clear [202], 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 [236]. Once revealed in 2018 by the Guardian [27], the Cambridge Analytica scandal generated strong public reactions and made the headlines all around the world [186, 2, 19]. The same year, Facebook's CEO was asked to answer questions in front of the US congress about Facebook's misuse of user data [182, 226]. 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

[41, 145, 219, 45, 114, 142, 90, 46] 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 [17] 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” [17], 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” [77]. On Figure 1.1 part ❶, we depict the appeal for surveillance as a negative influence on privacy.

Solove [200] proposes metaphors to better grasp the possible negative impacts of surveillance through literacy 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 [238] 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 [187] 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 [200, 187, 238] that surveillance organizations operate in 3 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 [187] 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 [238] 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" [225] and present their vision of privacy as "the right to be alone". The most influential definition of privacy in the policy world [17] 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 [229], 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 [135, 40].

To improve regulation, Rouvroy and Berns [188] 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 [136].

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) [214, 171, 30]. 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 [145].

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 [41, 145]. 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 [13]: *Sender-Receiver Anonymity*, *Sender Unlinkability*, and *Relationship Anonymity*. To define these properties, 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 to collect communication metadata. We refer to them as *Privacy Providers* as they act as an intermediary between the user and the service. VPN [2] providers pretend to prevent ISP [3] from spying on their users [22, 94, 211], proxy services circumvent state surveillance (like in China and Iran) [130, 174], whistleblower platforms enable employees to communicate with their organization to report internal frauds [82, 230, 210].

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” [170]. Even if not deceptive, these actors can be compromised [158]. Once the metadata are in the hands of a deceptive actor, users’ communication metadata face the same threats that pushed them 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 [36]. Over the last 40 years, three major designs with different privacy/performance trade-offs have been explored: mix networks [35, 126, 131, 132, 134, 169, 213, 217], dining cryptographer networks [80, 232, 43, 42, 44], and onion routing [37, 58, 69, 79, 84, 133, 191]. 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 [166]. 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 [131, 132]. 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 performances. 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 [166]. 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 [59]. 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 [228]. 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 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: 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 [172] and more generally any throughput intensive applications as Tor network bandwidth is limited. In response, we propose SAFE[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 SAFE 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 [65], 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 [25].

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 [36] 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 [35] 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 [131], Vuvuzela [217], Riffle [126], Loopix [169], Aqua [134], and Stadium [213], 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 relaying involved drastically multiply 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 [35]. 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 [80]. Consequently, numerous research works on DC-nets have emerged in recent years. Dissent [232, 43], Riposte [42], and Verdict [44] 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 [209], a simple proxy that strips origin 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 previous problems. The Java Anon Proxy [18] 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 alternatives designs were explored but suffered from structural limitations. PipeNet [48] suffers from trivial denials of service on the whole network. ISDN mixes [168] environment assumptions do not fit the current Internet topology.

The Onion Project [79] introduced the onion route design. It explored the different challenges involved with deploying an anonymity system: handling anonymous connections [177], configuring and integrating the system into the existing ecosystem [205], and security concerns [204]. In parallel, numerous systems were proposed where the whole infrastructure is contributed by users [69, 180, 179, 26, 181, 147]. Coined peer-to-peer systems, they are now regarded with caution [203] due to recurring attacks, especially on their relay discovery protocols [50, 206, 193].

Finally, the Onion Project [79] froze its final design with Tor [58, 203]: 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 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	~	✓	[131, 217, 126, 169, 134, 213]
DC-Net	~	X	✓	[80, 232, 43, 42, 44]
Onion Routing	✓	✓	X	[79, 177, 205, 204, 69, 180, 179, 26, 181, 147, 58, 203]

Table 2.1 – Anonymous network designs 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 a 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 performances 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 feature congestion and downtime. On Tor, these issues are concerning enough to be listed as an open research question [166] and generated many scholarly works to improve it [6, 157]. 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 transports (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 [50] (see Section 2.4), Tor announces all relays to all clients. Such a design leads to a bandwidth increase both when users and relays number 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 [148, 146, 123, 190]. Such works are required to be able to add new relays to the network but does not encourage people running 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* [89] 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 [103, 152]. Another design explores how contributing bandwidth to the network as a relay can be directly rewarded with higher priority circuits for the contributor [60, 152]. Finally, some designs introduce a currency to decorrelate resource providers and resource users [101, 102, 105]: 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 1:1 organic scaling: when a user joins, infrastructure capabilities scale accordingly. However, as seen before, onion routing has a poor experience with P2P systems [69, 180, 179, 26, 181, 147] 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 have not always full control of their network. As a response, Whisper [191] proposes solutions to enable nodes behind NAT to open circuits. Second, it 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 [148, 190]. Scholars have successfully proposed alternative designs for the directory to drastically reduce its complexity. Apart from the Tor directory scalability issue, it 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 relays 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 [33], Pairing Based Onion Routing [118], Diffie-Hellman Optimizations [161, 78, 12]. 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 performances 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, performances are 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 [31, 196, 3, 96]. However, performance variations are also due to queuing delays [57]. Path selection using relays performance history [199, 224, 15], global coordination [75], probing circuits on their creation [11, 96] 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 performances but do not protect from transient performance deterioration. 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 mul-

[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.

tiple mechanisms that can harm latency and throughput. Tor design results in the lack of end-to-end control flow which leads to the addition of a window-based control flow by 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 designs choice negatively impacts 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 [7] and lead to huge circuit queues inside relays before throttling the sending leading to high latencies in turn. AlSabah et al. [7] studied the problem and came with 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 [124]. Tuning the window enabled a small decrease in tail latencies but increased download time ; as a result authors deemed Tor's control flow mechanism as not very effective. Conversely, N23 should be the preferred approach according to authors as it enables a more important 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 bucket periodic refill occurs.

Historically refilled every second as a tradeoff between liveness and performance, it has been shown to harm latency, arbitrarily adding 2 seconds delay on some packets [57]. In 2012 a ticket [1] has been 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 [1, 121] converged to the value of 100ms.

A token-bucket can add as 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 passed queued inside the relay. The original round-robin mechanism is known [59] 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 [86], Tschorsch and Scheuermann [212] 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 punctually transfer data. In this context, resource allocation must not be fair at a given time but over time. To satisfy this constraint, researched have proposed to prioritize circuits according to an EWMA indicator of their bandwidth usage over time [207] or more advanced machine learning techniques [4]. Other heuristics were proposed: KIST [107] authors 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. KIST [107] authors 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 [106].

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

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 [153] 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 latencies spikes.

One observation made by scholars is that non-interactive transfers, like file transfers, impede interactive ones, like VoIP. As a solution, Torchestra [81] authors propose to open two TCP links between each relay: one for non-interactive traffic, one for interactive one. TCP-over-DTLS [176] and PCTCP [5] 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 [74] proposes an improvement on Torchestra [81] while pointing that opening one socket per-stream [176, 5] 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 do not require that packets are sent ordered between each relay, only between the sender and the recipient. Following this observation, UDP-OR [218] proposes to forward packets on UDP transport between relays and operate a TCP sockets only end-to-end. Pushing the idea further, scholars proposed end-to-end abstraction change to better fit different communication needs: UDP [139], unordered TCP [159], and QUIC [164].

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

Multipath Having a multipath approach to the Tor network opens us to new perspectives to leverage available relays. Recognizing its usefulness for available and real-time communication, MPTCP is being standardized as an RFC [88]. Some works focused on tuning MPTCP specifically for low latencies: by duplicating data on two links [73] or by probing links regularly and schedule traffic on the fastest one [72]. 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 [129] 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 [115], Conflux [8], mTor [235], and mUDP-OR [53] were proposed. MPTCP Tor, Conflux, and mTor simply aggregates 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 [218] 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 particular 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 [97], 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 performances 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 [133, 132] 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 [133] 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 [132] 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 [100]. 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 [116, 64] explore the portage of SIP infrastructures on Tor. The main principle of their work is to preserve privacy in the SIP signaling protocol but 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 [76] improves latency by duplicating traffic over two onion links similarly to ReMP [73]. 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 [59]. 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 [143].

As an alternative, OnionShare [160] integrates well with Tor and preserves anonymity properties. While it does not leak data, a wider adoption would lead to Tor network collapse. 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 4 properties: `Signaling`, `Latency`, and `Throughput` improvement 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.

	Signaling	**Latency**	**Throughput**	**Deployability**	**References**
More Relays					
Directory	✓	X	X	✓	[148, 146, 123, 190]
Incentive	X	✓	✓	XX	[103, 152, 60, 152, 101, 102, 105]
Cost	X	✓✓	✓✓	X	[69, 180, 179, 26, 181, 147, 191]
Better Circuits					
Construction	∼	X	X	∼	[33, 118, 161, 78, 12]
Selection	X	∼	∼	✓	[31, 196, 3, 96, 199, 224, 15, 75, 11]
Better Relays					
Control flow	X	✓	X	∼	[7]
Rate limit	X	✓	X	✓	[57, 1, 121]
Scheduling	X	✓	∼	✓	[212, 207, 4, 107, 106]
Better Transport					
Single path	X	✓✓	X	XX	[81, 176, 5, 74, 81, 218, 139, 159, 164]
Multipath	X	✓✓	✓	∼	[88, 73, 72, 129, 115, 8, 235, 53]
Applications					
VoIP	X	∼	X	∼	[133, 132, 116, 64, 76]
File transfer	X	X	X	✓	[160, 59, 143]

Table 2.2 – Onion routing optimizations comparison

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 can be hardly ensured 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 suffers 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 are simply built with a central server multiplexing all connections of the group members. Such solutions are far from being satisfying in term 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 [56], 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 [122]. 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 [63]. 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 [162] uses a pull protocol to learn new data in a dynamic network. Coolstreaming [137] uses a pull protocol to fetch data where 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 [189] proposes a solution to discover the missing messages in the pull phase but works only with a single source. Instead of ordering messages, Pulp [65] 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 [34, 189] or Random Linear Network Coding [38] but need encoding at the source or ordering message, which limits these techniques to single-source scenarios. Theoretical bounds have also been studied for multi-sender scenarios [54, 85] 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 [67, 119] 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.

	Latency	**Throughput**	**Applicability**	**References**
Push	✓	X	✓	[122, 63]
Pull	X	✓	✓	[162, 137]
Push/Pull	~	~	✓	[189, 65]
RLNC	✓	✓	X	[34, 189, 38, 54, 85, 67, 119]

Table 2.3 – Gossip designs comparison

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 performances. Our goal is to make RLNC algorithms usable in the same situations as Push and/or Pull algorithms and therefore tick the `Applicability` property. Once gossip is cheap enough, we can consider run 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

Optimization discussed in the previous chapter could come in tension 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 [29] introduced concepts like anonymity, unlinkability, undetectability, unobservability, etc. Such terminology has been revisited [13] 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, 195, 111, 184]. 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 [195] but also by making an intersection between Tor users set and some exterior sets [231]. In the long run, with relays rotation, only 2 malicious relays in the whole network are needed to compromise users' anonymity [234, 233]. 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 [222].

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 [149]. 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 [28, 175]. 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 [125]: such information can be used later when conducting a website fingerprinting attack [163]. By connecting to an Onion Service, it is also possible to infer its guard by generating a traffic signature [21]. Some of these attacks can be mitigated through Onion Services version 3 [144], 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 attack is possible on networks that only advertise a subset N_{subset} of all network relays N_{network}. Such an idea 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 [49, 50]. 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 [169, 213, 217, 133, 132]. 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 [155, 140]. 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 3.5**. We use VoIP traffic emulation as well as the off-the-shelf `gstreamer` [83] VoIP client using the OPUS [39] 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 [76], 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) [98, 100, 99]. 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) [185] 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) [194] is used to transmit VoIP

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

Table 3.1 – VoIP network performance requirements, following the recommendations of the International Telecommunication Union (ITU) [100] and applying them to the OPUS codec [216, 215].

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 [216, 215]. It targets interactive, low-delay communication and computational efficiency. OPUS has been consistently ranked in comparative studies as the highest-quality audio format for low and medium bitrates [93, 113]. We emphasize that our analysis would be similar for other open codecs, e.g. the Internet Low Bit Rate Codec (iLBC) [9] or Xiph.Org Foundation's former codecs Vorbis [14] and Speex [91].

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 [99]. We adopt the same goal and consider, conservatively, a duration of calls of 90 minutes, derived from the maximum call durations at major representative carriers [95, 220].

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 [93, 39] 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] [208]. Han *et al.* [87] 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%.

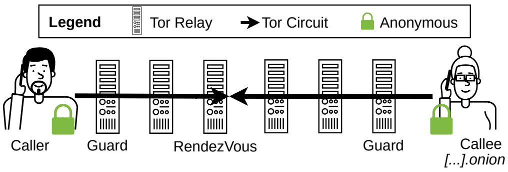
Third guarantee: *interactive conversations.* In addition to an interrupted and good-quality voice signal, users of voice calls expect to be able to exchange information interactively, e.g., be able to seamlessly synchronize on when to stop and start talking in a conversation.

Interactivity primarily depends on latency [192]. The ITU published recommendation G.114 [100] 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 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) packetization and (2) buffering. Each frame is encapsulated in a new packet every T_{frame} ms. 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, Kurose, and Towsley [151]. In this model, all frames are delayed to the maximum or n^{th} percentile of observed latency and we allow frame drops. Moon, Kurose, and Towsley [151] and others [120, 138] have proposed jitter buffer implementations performing close to this theoretical optimum. Therefore, we consider T_{jitter} as negligible. Finally, as we allow a 1% frame drop we consider the 99$^{\text{th}}$ latency for our mouth to ear delay constraints.

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



Icons by RROOK and Juan Pablo Bravo from the Noun Project

Figure 3.1 – Structure of a Tor link with onion services.

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 [58] 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 anonymity for both parties through the notion of *Onion Services*. Figure 3.1 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.

The setup of an onion service involves an *introduction-point* relay, whose identity can be retrieved from a DHT by using the *onion address* of the service. We note that the introduction point is not used for the actual exchange of data. As a result, the rendezvous points for different onion routes to the same destination will be different.

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 512-Byte *cells* [155, 140]. When a

[2] While Tor advertises using $n = 1$ by default, it effectively uses $n = 2$ [165].

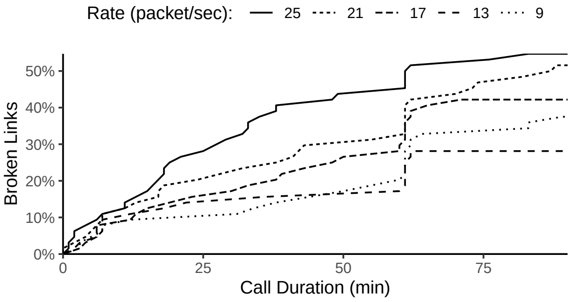


Figure 3.2 – Failed Tor links over time.

packet being transmitted over a Tor connection is less than 512 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 [36, 217, 213]. 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 [183, 92].

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 use two separate instances of the Tor client in its default configuration, and create circuits independently. All reported experiments were conducted in January 2020.

Connection stability. We first evaluate the reliability of Tor onion links over our target call duration (90 minutes). We send packets at varying frequencies, from 9 to 25 packets per second. The highest frequencies of 17 and 25 packets per second correspond to frame

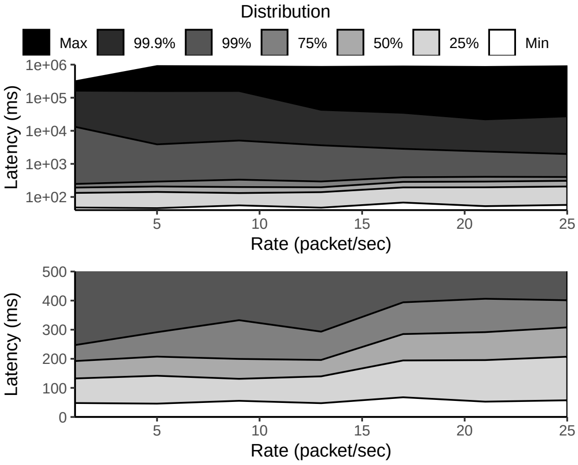


Figure 3.3 – Latency distribution for varying traffic rates.

rates supported by the OPUS codec (T_{frame} =$\{60, 40\}$ ms, respectively). The lower frequencies allow us to understand the evolution of stability with lower traffic. Figure 3.2 reports the cumulative rate of failed links (i.e., for which packets are no longer transmitted) as a function of time. Each plot aggregates measurements over 64 different onion links. At 25 packets per second, 40% of the links fail within 40 minutes, and 6% fail within the first 2 minutes. At 17 packets per second, these numbers drop to 22% and 1.5%. More generally, we observe a correlation between the risk of link failure and the sending rate: higher packet rates result in higher failure rates[3].

Variability of one-way latency. Figure 3.3 presents the distribution of the observed one-way latency for 64 measured onion links. Note that the distribution is split into low values using a linear scale, and high values using a logarithmic scale. Each line represents a distribution with stacked-up percentiles (e.g. 25% of the measurements are below the value). Even with the lowest sending frequency, median one-way latency is above the ideal latency of 150 ms. With the highest sending rates ($\geq$17 packets per second), more than 25% of the packets exceed the 400 ms of acceptable latency.

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 from a large number of poorly performing links with a few, identifiable, good links, or if any link can experience periodic latency bursts. Figure 3.4 presents the one-way-latency distribution for each of the 64 links, ranked by median latency (top) or max latency

[3] We presume this is a result of Tor scheduling [107, 104] and anti-DDoS [51] policies. An in-depth analysis is, however, outside the scope of this thesis.

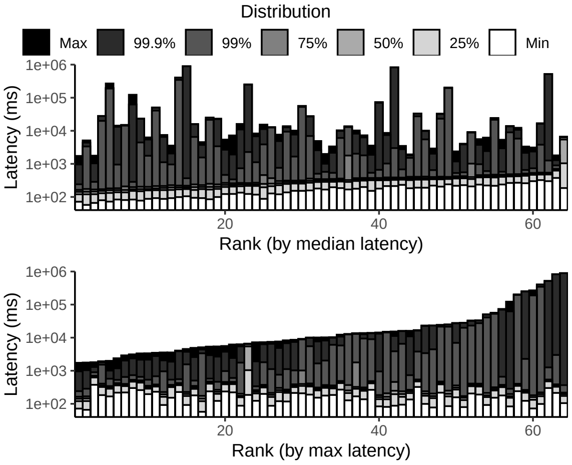


Figure 3.4 – Tor links' latency distribution at 25 pkt/sec ordered by median (top) and max (bottom) latency.

(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 75th percentile) [4]. We refer to these high latency periods as *latency spikes* in the rest of this chapter.

Discussion. Our experiments confirm the general unpredictability of the performance of Tor links. Due to Tor's exclusive support for TCP [5], 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 packet rate correlates with the probability of networking problems: higher packet rates 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 make the case for using multiple links, benefiting from periods of good performance, and quickly switching links when experiencing latency spikes.

[4] This unpredictable performance is confirmed, in fact, by a blog post by the Tor project [166]. 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.*".

[5] 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 [150].

3.3 Donar: Enabling VoIP over Tor

We now present Donar, our configurable and versatile solution for high-quality VoIP over Tor. The design of Donar is driven by the requirements set in Section 3.1, but also by the need to minimize the load on the Tor network.

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 communication without leaking metadata about communicating parties.

The foundational principle of Donar is to use multiple Tor links to enable *redundancy* and leverage *diversity.* Using multiple links introduces additional state costs for Tor relays, which have to maintain more information on their routing tables. A distinctive feature of Donar is, however, that in its default configuration it does not incur *any* additional bandwidth cost compared to the use of a single link. We determine in our evaluation (§5.4) that opening 12 onion links using the Tor client, and actively using half of these for transmitting VoIP frames, enables Donar to consistently meet its objectives at a reasonable cost.

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 512 Bytes. Donar leverages, instead, this space to re-send the previous frame without changing the necessary bandwidth requirements [6]. 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 comple-

[6] 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.

mentary 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 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 512-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_{1ST} (first-class) group contains the n_{1ST} *fastest* links. The L_{2ND} (second-class) group contains the n_{2ND} following links. Typically, we use the same number of links in the two groups, i.e., $n_{\text{1ST}} = n_{\text{2ND}}$. Finally, the remaining $n_{\text{INACTIVE}} = n_{\text{LINKS}} - n_{\text{1ST}} - n_{\text{2ND}}$ 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_{1ST} 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_{2ND} 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).[7] 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 $n_{\text{LINKS}} = n_{\text{1ST}} + n_{\text{2ND}} + n_{\text{INACTIVE}}$ links and assigns 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_{2ND} 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_{2ND} group and demotes it to the L_{INACTIVE} group. In return, it promotes to L_{2ND} 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_{1ST} and L_{2ND} 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_{1ST} and L_{2ND} 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 [76]'s Duplication mode, which systematically sends VoIP packets onto the same two links.

[7] 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.

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 [64, 116]. Furthermore, with the exception of the audio codec negotiation, SIP functionalities largely overlap mechanisms already offered by Tor [64, 116].

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 [144], 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 [58, 155] to perform traffic-correlation attacks [234, 112]. 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 [234] 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 [233] 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 [13], sender/recipient anonymity refers to the ability to hide one endpoint's identity from the other. As discussed in [233], 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 [222].

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 [149, 125, 163, 28, 175] or use a fake DONAR client and perform timing attacks [154] 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 [144]. Finally, while several authors have shown that an adversary could locate onion service endpoints even when they were not publicly advertised [21, 125, 163, 149], 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 [112]. 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

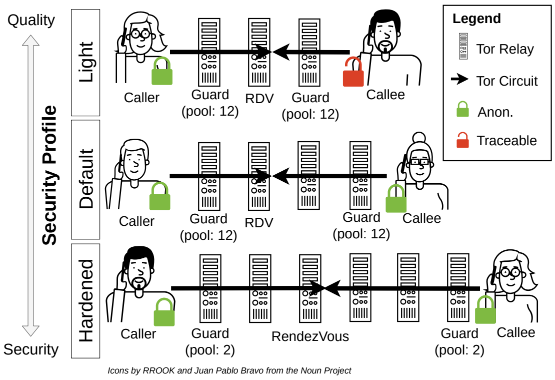


Figure 3.5 – Security configurations.

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.

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 3.5. 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 *Hardened* 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.[8] The *Default* configuration sets up the Tor client so that each created link has: (i) 12 guards on each side, i.e. each link uses a different guard, (ii) two fewer Tor relays compared to Tor's default link settings. Finally, the *Light* configuration further reduces the number of relays leading to the use of a *single* Tor relay (the guard) 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's use of more guards to improve path diversity increases the attack surface. Consequently, only the *Default* and *Light* configurations expose DONAR to increased risks with respect to standard

[8] Even if Tor's documentation discusses using only one guard, the default client uses two.

Tor usage. Additionally, compared to the *hardened* configuration, the *Default* (resp. *Light*) one reduces the number of relays in links by two (resp. three). 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 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 [16]. These observations motivate our choice of 2-relay circuits with better latency in our Default configuration.

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

The DONAR proxy interfaces a VoIP application with the Tor client. [9] 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 $n_{\text{1ST}} = 3$ links, $n_{\text{2ND}} = 3$ links,

[9] The Tor software is evolving quickly, especially considering the recent V3 of Onion services. To benefit from latest bug fixes, we followed Tor master branch and used commit ff931334 for our experiments.

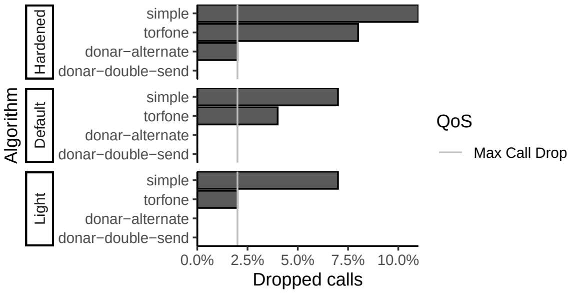


Figure 3.6 – Dropped calls after 90 minutes for SIMPLE, TORFONE, and DONAR setups.

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 (Hardened, Default, or Light). 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 [76]: 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 3.6 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 hardened configuration. TORFONE only meets the goal in the Light 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 3.7. Our mouth-to-ear latency objective is 150 ms, and our limit is 400 ms. As T_{frame}=40 ms, we wish network

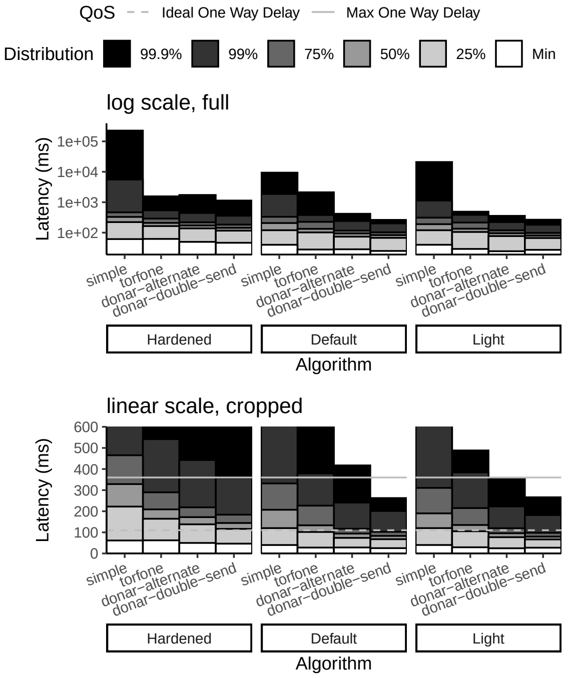


Figure 3.7 – Latency comparison between SIMPLE, TORFONE, DONAR ALTERNATE and DONAR DOUBLE-SEND.

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

All DONAR variants except once again for the most conservative variant of the ALTERNATE policy with the Hardened configuration, satisfy the requirement of a 99$^{\text{th}}$-perc. latency of 360ms. In contrast, when using the Default and Light configurations, 99$^{\text{th}}$-perc. latency remains under 250 ms. This confirms the ability of DONAR to provide good-quality anonymous calls over Tor with the same bandwidth consumption as SIMPLE, which experiences 99$^{\text{th}}$-perc. latencies over 1s even in the Light configuration. The DOUBLE-SEND policy achieves better performance at the cost of doubled bandwidth, matching (albeit closely) the requirement even for the Hardened configuration. For the same bandwidth consumption, TORFONE only provides this guarantee under the Light configuration.

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$^{\text{th}}$-perc. latency, and count the number of calls that systematically meet latency

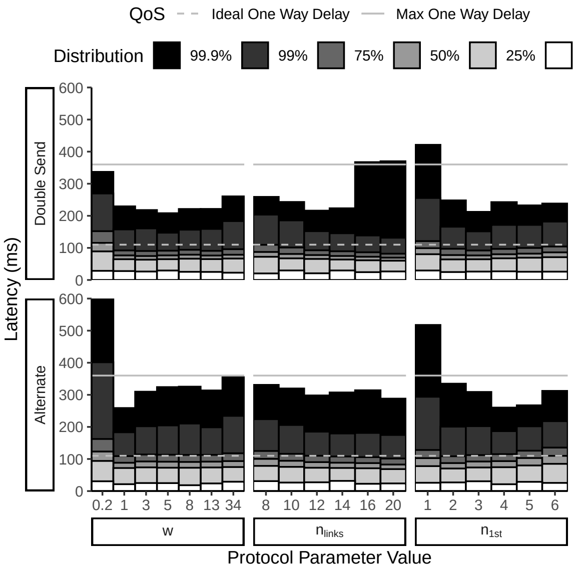


Figure 3.8 – Impact of protocol parameters (w, n_{LINKS} and $n_{\text{1ST}} = n_{\text{2ND}}$) on frame delivery latencies.

requirements out of the 64 experiments done for each configuration. Our results confirm that Donar is able to meet the 360 ms latency threshold for all experiments in all configurations, with the exception of the alternate policy under the Hardened configuration. We also confirmed empirically the results obtained under the Default 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_{1ST} (we use $n_{\text{1ST}} = n_{\text{2ND}}$). In the experiments reported in the previous section, we employed the default values of $w = 2s$, $n_{\text{LINKS}} = 12$ and $n_{\text{1ST}} = n_{\text{2ND}} = 3$. We detail in the following how we selected this default configuration.

We present in Figure 3.8 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 promotes links that are stable over time. We can observe on the left side of Figure 3.8 that the best value of w for the DOUBLE-SEND policy is 5s, while the best for the ALTERNATE appears to be 2s. We 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 $n_{\text{1ST}} = n_{\text{2ND}}$ 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_{1ST} 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 $n_{\text{1ST}} = 1$ yields high latencies with either variant, while $n_{\text{1ST}} = 3$ or $n_{\text{1ST}} = 4$ offer a good compromise. We choose $n_{\text{1ST}} = 3$ 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 3.9, 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. We recommend, therefore, that users wishing a high level of security employ the DOUBLE-SEND policy together with the Hardened configuration.

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 3.10. 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

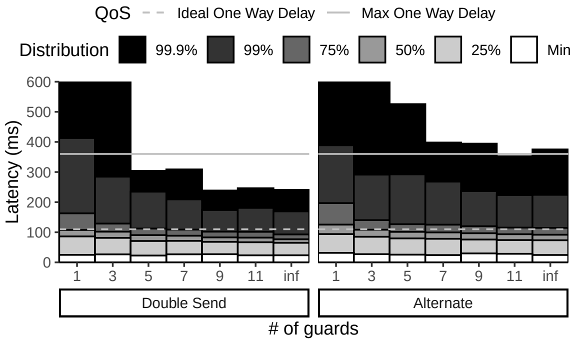


Figure 3.9 – Impact of Tor guards number on latencies.

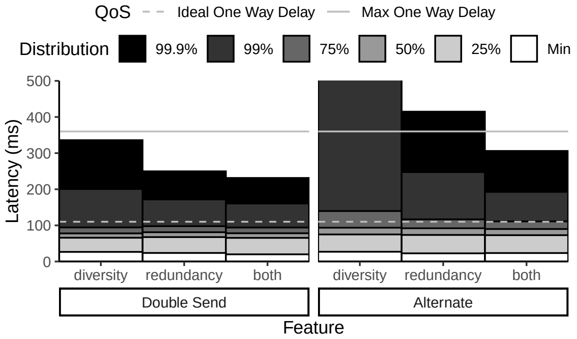


Figure 3.10 – Diversity & redundancy complementarity.

time, and whether this first delivery concerns a primary or a duplicate copy. The first 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 $L_{1\text{ST}}$ group arrive first, and only 6% are received as a duplicate copy via an $L_{2\text{ND}}$ link, despite being sent 40 ms later. When the primary frame copy is sent over an $L_{2\text{ND}}$ link, however, only 48% arrive before the duplicate copy sent over an $L_{1\text{ST}}$ 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 $L_{1\text{ST}}$ link, 14% are received as a primary copy on an $L_{1\text{ST}}$ link, and only 13% are received as a duplicate copy. Using $L_{2\text{ND}}$ 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.

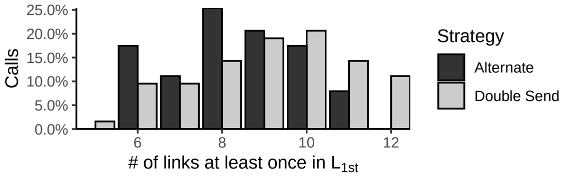


Figure 3.11 – How many links were L_{1ST} at least once?

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 3.11. 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 3.12. This is the latency that

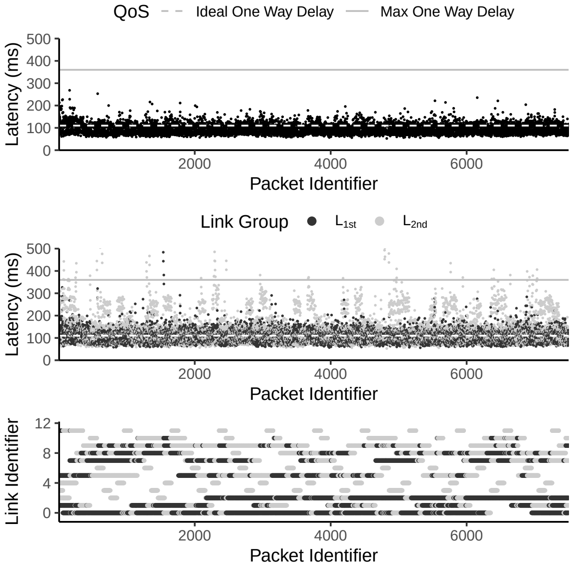


Figure 3.12 – Stability over time.

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.

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 250ms 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 Safe, a new anonymity solution enabling secure file exchanges that organically scale albeit built on onion routes. Safe leverages edge devices to drastically increase network throughput by being designed at its heart to provide good performances with relays characterized by heavy churn and volatility. Safe implements its own onion routing schemes to offer continuous availability and better network throughput. In particular, Safe introduces the concept of Multiple Onion Circuit (`MOC`). Safe selects multiple relays instead of one at each hop. Concretely, when receiving a packet, a relay will query its internal circuit routing table to find potential next hops and select the one that is up and less congested.

Safe preserves anonymity: the negative impact of multiple relays per hop is compensated by the greater number of relays available and usable in the network. More specifically, the number of relays per hop can be adjusted to maximize anonymity according to the desired circuit's availability probability. Especially, we show that for a relay set that features a lot of churns it is better in terms of anonymity to put multiple volatile relays per-hop than ignoring them.

Our contributions are as follows:

— We introduce a novel onion routing protocol that (i) leverages multipath onion circuit, to provide continuous availability and better throughput (ii) leverages prediction, to preserve anonymity by picking the right amount of relays.

— We integrate our protocol in a global system that offers anonymous file transfer. Our system is built following the well-studied Hidden Service system, popularized by Tor Onion Services.

— We evaluate our system in a simulation and with a real deployment in various configurations of churn. We demonstrate that introducing new paradigms in onion routing protocols opens new applications. Compared to Tor Onion Services, we show that we can scale a file transfer solution while the infrastructure cost is supported by the users.

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 SAFE in section 4.5. We conclude this chapter in section 4.6.

4.2 Problem Definition and Goals

4.2.1 Problem definition

In Tor, relays are designed to be run mostly in the core network (Figure 4.1①). However, deploying relays in the core network requires to provision dedicated and costly servers. For instance, for deploying a relay, the Tor community highly recommends a server with at least both 16 Mbit/s of upload and download bandwidth with very high availability, i.e. an uptime of 24/7. Additionally, to be resilient to attacks and outages, particular care should be taken to rent servers from different providers so as servers are located into different Autonomous System (AS) to provide network diversity, preventing massive renting of servers from the same provider. As a result, in Tor, there are 300× more users than relays, putting a bottleneck on the network throughput. We argue that relays must be instead designed to be mostly deployed at the edge, where resources are numerous and already provisioned thanks to the massive user base (Figure 4.1②). In a more global manner, we argue that the data plane should be provisioned entirely at the edge.

Running relays massively in the edge network raises new challenges as edge devices feature high volatility. Volatility is induced by numerous causes, ranging from power failures to users switching off their devices, implying the notion of *availability*. Availability has inherently a strong impact on the reliability at the application level. For instance, to securely and anonymously share files over Tor between two users, `Alice` and `Bob`, OnionShare requires to build a circuit made of 6 Tor relays to transfer data between the

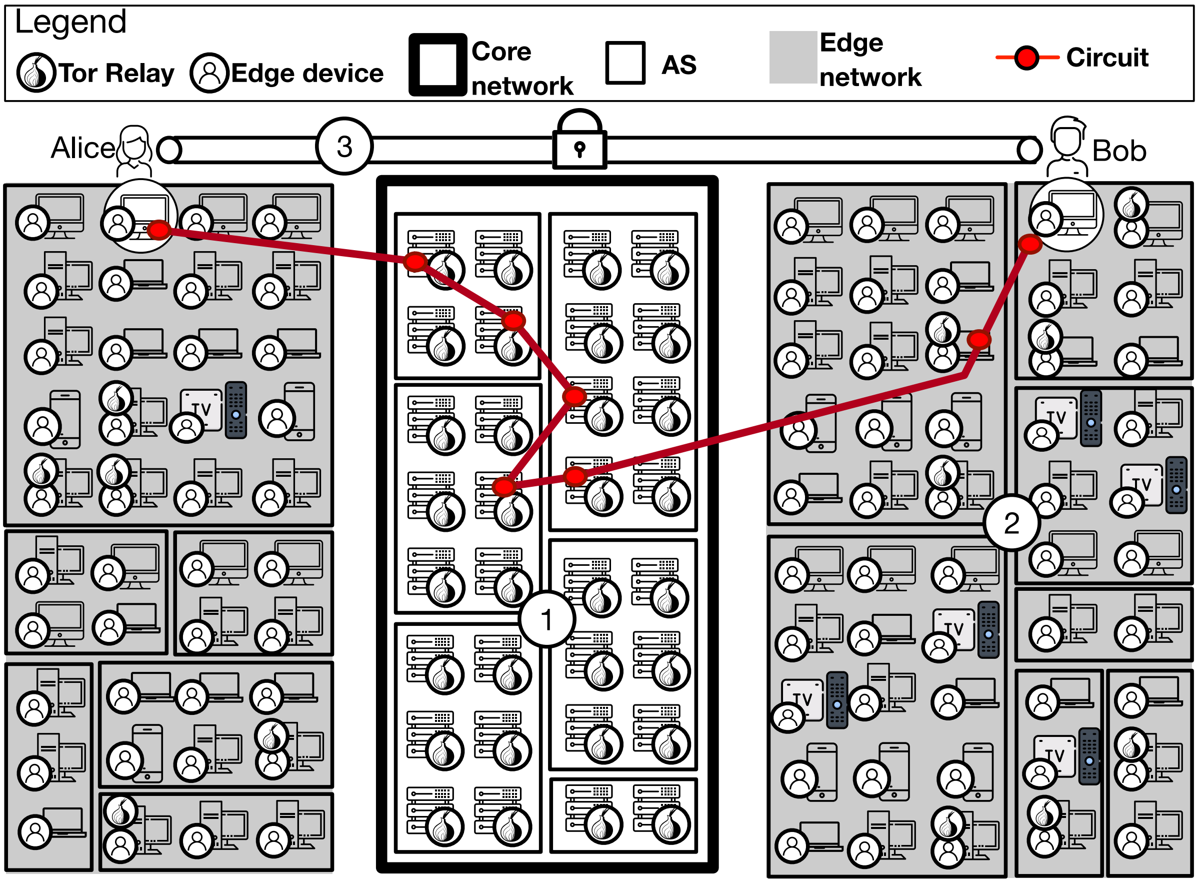


Figure 4.1 – Relays in the Tor core network

sender and the receiver (Figure 4.1③). However, considering that relays in the edge are volatile, it strongly implies that we can no longer suppose that all relays of an onion circuit will remain up during the whole file transfer causing unpredictable timeout and errors.

To discern the current proportion of Tor relays into either the edge or in the core and its related magnitude of the problem of availability, we have collected from Tor metrics all Tor consensuses over a year (published every hour) during the 2019-05-01 to 2020-05-31 timespan. It results in a dataset composed of 32695 identified relays. However, collected metrics do not allow us to make the difference between relays that were definitely removed from the ones that are only unavailable for a short period of time. To avoid our data to be polluted by relays that only appear once over a year (e.g. relays deployed only for testing purposes), we removed relays whose lifetime is inferior to one week. We define the lifetime of a relay as the time elapsed between its first and last appearance over one

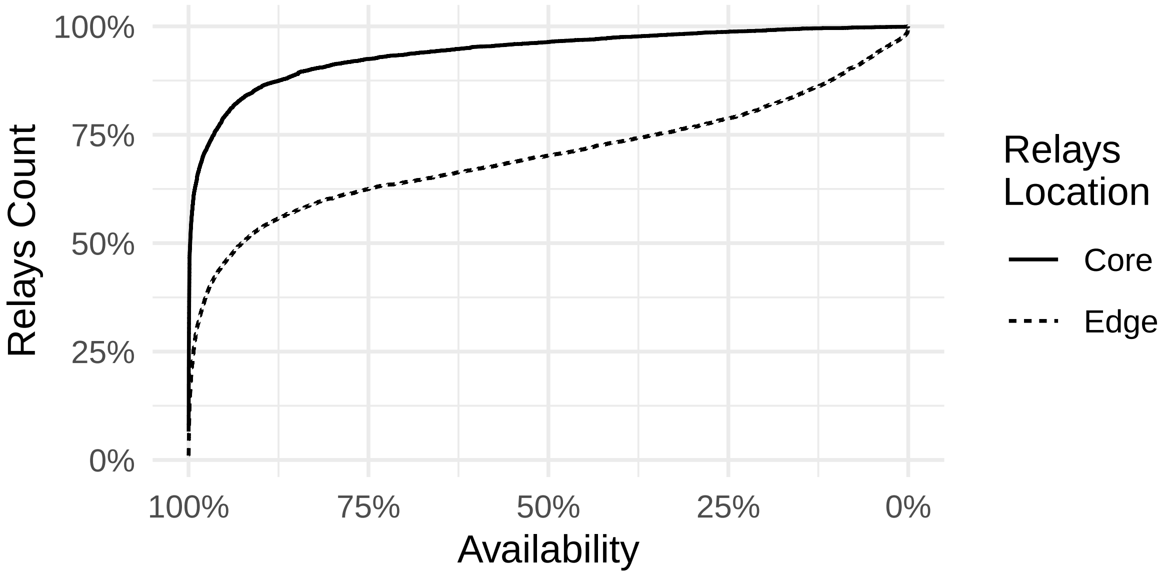


Figure 4.2 – Empirical Cumulative Distribution Function of Relays Availability registered in the Tor relay directory. Data were collected between 2019-05-01 and 2020-05-31.

year. Correspondingly, our dataset counts finally 19585 relays. However, only a median of 6432 relays is available at a point in time. We further use IPHub [173] that indexes IP as residential or datacenter, to determine if relays are either in the edge or in the core network. Correspondingly, over 6432 relays online, 2522 are flagged as being in the edge, whereas 3910 are flagged as being in the core network. Finally, Figure 4.2 gives the Empirical Cumulative Distribution Function (ECDF) of relays availability depending on whether they are in the edge or in the core network. It appears that $62,5\%$ of nodes in the core network are 100% available as opposed to relays in the edge that are only 25% available at 100%, highlighting the consequence of being at the edge.

Nevertheless being available is not enough to be selected in a Tor circuit. In the last decades, massive research works have been done to refine relay selection algorithms. Multiple criteria such as availability, latency, bandwidth, congestion, security, and/or location are involved to select relays [198, 221, 196, 223, 62], reducing chances for relays in the edge to be selected. For instance, the first node of each extremity of a Tor circuit, commonly named guards, are carefully selected notably to protect users from predecessor attacks and denial of service. As a matter of fact, in our dataset, over 2522 relays in the edge, only 736 of them have been effectively considered as guards by Tor. Comparatively, 2272 over 3910 relays in the core network have been used as guards.

Promoting an anonymous file transfer service requires a huge amount of bandwidth that Tor, in its current design, is unable to absorb. Increasing drastically Tor network throughput inherently implies consequently increasing the number of Tor relays in the core network. However, it is unreasonable to put the burden on a small part of users to invest in always more relays. To counterbalance this phenomenon, we argue to massively deploy relays in the edge instead of the core networks, i.e to involve all users. In other terms, to make relays at the edge be the norm, it is required to revisit the onion routing scheme to overcome the related availability issues coming from this design choice while providing similar privacy properties as Tor.

4.2.2 Functional and performance goals

Decentralized. File exchanges should not depend on the availability of a specific cloud service and should remain *decentralized.* Decentralization has several advantages. First, it prevents a service provider from "denylisting" a specific user from using the service, making censorship attacks more difficult [1]. Second, it enables better scaling and can reduce significantly the operational cost of the service. We enact this reduction by leveraging edge resources to *assist* dedicated core resources in providing the service.

Highly available. The file exchange service must be offered despite the uncertainty inherent to the availability of constituents of the system. There should be no single point of failure. In particular, the lower general availability of edge assisting servers should not impact the success of file transfers, and the system must transparently deal with churn, faults, and disconnections.

Load balanced. It is further important that core and edge resources be used fairly, i.e., that the amount of transfer that each supports be equivalent to a fraction of its capacity. This is particularly important for edge servers, in order for contributing users to pay a fair share of their resources to the service and avoid service degradation.

Performance. Finally, transferring large files should not take extensive amounts of time, and performances should be equivalent to `WeTransfer`, `Smash`, or to that of personal cloud storage solutions supporting sharing features such, as `Dropbox` or `Mega`. Ideally, transfers should be capped in practice by the upload capacity of the file sender.

[1] A distributed service running over multiple cloud or edge data centers but under control of the same administrative domain remains a single cloud service from the perspective of censorship

4.3 Approach

We now introduce Safe, an anonymity network that leverages edge devices to enable anonymous file transfers, and more generally any anonymous throughput intensive application.

4.3.1 Overview

The Safe infrastructure, in a way similar to Tor, reuse the following key concepts: (i) relay directory, (ii) service directories, (iii) servers, (iv) users, and (v) onion circuits. Where appropriate, Safe reuses, as is, of building blocks of Tor.

Relay directories. Safe reuses the concepts of Tor Directory Authorities (DA) for relay directories to collect the complete information about each existing relay in the system. Regularly, the list of existing relays is updated among relay directories following the Tor consensus protocol. The global view of the network may be seen as a scalability bottleneck. However, such a bottleneck is orthogonal to our approach, and Safe would leverage for an implementation existing research work enabling users to discover relays without maintaining a global state view of the network [190, 123].

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

Servers. Safe distinguishes two types of servers: *Dedicated servers* and *edge servers.* Dedicated servers are contributed by different organizations to host key building blocks of Safe (e.g. relay and service directories), and run in the *core* of the network, e.g., on public clouds such as Amazon EC2 or Microsoft Azure, similarly to Tor. They offer good connectivity and are relatively stable, but their number is limited as provisioning a new server induces operational costs for its owner. As a result, their aggregated bandwidth capacity remains limited and difficult to scale. Edge servers are devices with limited capabilities contributed by users. In Safe, relays, and potentially all of them, are deployed on edge servers. These servers can be set-top boxes or small smart environment hubs, whose capacity can be used when not required for their primary function. For instance, a set-top box must focus on provisioning broadband internet access and high-definition TV when a family is home but remains otherwise unused during most of the day and night. The same applies to a home server implementing household management automation and has spare bandwidth and CPU capacity. The general availability of these edge resources is therefore generally *lower* than that of dedicated servers. Their number, on the other hand, is supposed to be massively greater than the number of dedicated core servers and, while their individual capacity is generally smaller, their aggregated available bandwidth is much larger than that of all dedicated servers combined. In addition, as these resources are contributed by users themselves, we expect their number to gracefully scale with the popularity of the system.

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

Onion circuits. SAFE differs from Tor in the way onion circuits are created and managed. SAFE's aim is to overcome the volatility of relays massively located by design in the edge network. Specifically, SAFE adds redundancy by packing multiple relays to each hop in an onion circuit, resulting in a Multipath Onion Circuit (MOC) as depicted in Figure 4.3 ①. Using a MOC, a circuit remains functional as long as at least one relay remains up per hop, increasing so the robustness (Figure 4.3 ②). Comparatively, regular onion circuits, as used in Tor, are down as soon as one relay lacks availability, leading the client to recreate a new circuit Figure 4.3 ③. In SAFE, intermittent disconnection of relays inside a MOC does not require clients to both recreate or reconfigure its currently used circuit. In a MOC, paths are independent of each other, and disconnected relays can reopen connections with relays of other hops seamlessly. Finding the right amount of relays per hop in a MOC is a trade-off between security and reliability. To have an accurate security analysis, we set a fixed number of relays per hop. Relays in a hop are selected according to both an availability threshold and its congestion state. As long as the availability threshold for a hop is not reached, the relay with the lowest availability is discarded, and a new random one is picked.

The MOC mechanism enables SAFE to provide more robust onion circuits with increased throughput. The network consumption is further optimized as, in each hop, each relay selects relays of the next hope that are less congested (amongst available ones) to forward its data. In the following, we describe in detail our own implementation of onion circuits with our MOC principle.

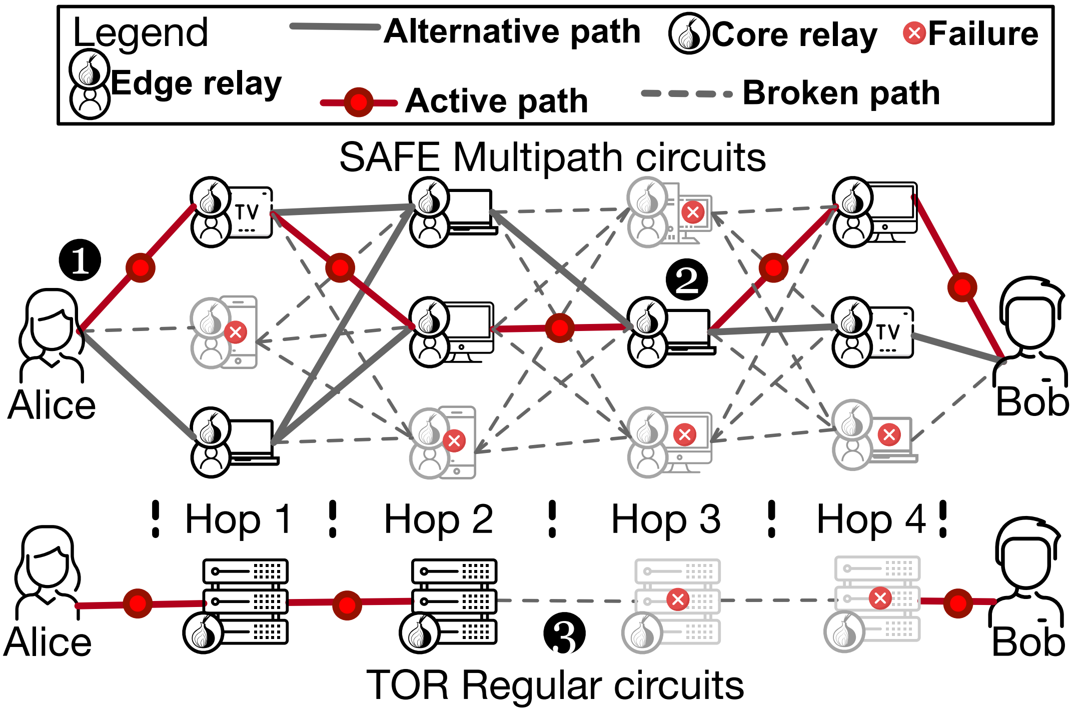


Figure 4.3 – SAFE introduces Multipath Onion Circuits (MOC). MOC are robusts as multiple relays are packed per hop and, in the long run, connections can be restored by relays themselves. MOC are also efficient, as each relay can choose the less congested relay of the next hop to maximize network usage.

4.3.2 Relaying Data In Multipath Circuits

Onion routing consists of recursively encoding a datagram starting from the receiver key, then the last-hop key, and so on, to the first-hop key. The resulting data is referred to as an onion cell. Onion routing can be done in a stateless manner by using relays public keys to recursively encrypt the payload. Next relay address must be embedded on each layer for each relay. However public key cryptography is slow and routing information waste bandwidth. To overcome this limitation, systems like Tor introduced switched virtual circuits (i.e. onion circuits), where, for hop i, next relay R^{i+1} and session key K^i are sent during circuit opening and associated with a circuit identifier I^i, constituting the tuple (I^i, K^i, R^{i+1}) persisted in relay circuit table. Accordingly, a data cell needs only to contain the circuit identifier I^i to be decoded, as (K^i, R^{i+1}) can be retrieved from the relay circuit table. In SAFE, we introduce multiple relays per hop, modifying routing information that must be stored by a relay to $(I^i, K^i, [R_1^{i+1} \cdots R_n^{i+1}])$ where n is the number of relays per hop.

Safe, like Tor, uses *control cells* to manage (i.e create, update, destroy, ping) `MOC`, and *relay cells* to carry end-to-end data. Although a relay has several alternatives to forward its *relay cells*, a relay transmits them only once at the next hop. Indeed, between its decoding and forwarding steps, a relay must choose the next candidate to forward relay cells. On receiving data, a relay peeks one incoming relay cell from its receiving sockets, reviews all sending sockets associated with it to select the best next hop. The rationale for the selection of the candidate is that it must be available and not congested. To this end, Safe leverages on TCP sockets: availability is estimated from the candidate socket not being broken *via* heartbeats and timeout, while congestion is estimated through the length of Safe internal sending queue for each candidate of the circuit (i.e. the candidate with the smallest queue is the less congested). The internal sending queue of a candidate grows if the relay receives packets faster than it is able to deliver them to all its candidates. If all candidate sending queue length exceeds a defined threshold, Safe stop reading incoming packets for the considered `MOC`, waiting that one of the candidate sending queue size decreases. 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. Finally, as a background thread, a Safe relay is in charge of maintaining connections alive, including retrying broken connections.

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

4.3.3 Multipath Circuits Lifetime

Prior to sending data, a circuit must be created. Accordingly, a descriptor must be sent to each relay *via* control cells, enabling to inform relays of the existence of the circuit (i.e. to populate relays' circuit tables). Contrary to relay cells, control cells are sent and relayed to all relays of the next hop, enabling each of them to set up their circuit tables. However, a relay forwards control cells only if they are new, (i.e. a new entry into its circuit table). Consequently, relays will receive multiple times the same control cell, at max n times if all relays of the previous hop are up, and if the descriptor is new. Additionally, to allow users to send data in both ways, and then to provide full-duplex circuits, previous-hop relays R_j^{i-1} are added to descriptors. Hence, the full descriptor D^i that is sent for hop i by Safe looks like $(I^i, K^i, [R_1^{i+1} \cdots R_n^{i+1}], [R_1^{i-1} \cdots R_n^{i-1}])$. Descriptor D^i must be then sent to hop i relays, thus it needs to be decipherable by any of the current hop i's members, and only by them, requiring so the use of a broadcast encryption scheme.

Broadcast Encryption. SAFE leverages Broadcast Encryption (BE) [66] to share descriptors among all nodes of the same hop. Specifically, SAFE derives its encryption process from Hybrid Encryption [201] (as used in PGP), where a message $\mathcal{M}$ is encrypted into a cipher $\mathcal{C}$ using a unique symmetric key k (e.g. using AES). Each member of the hop i must be given this key, which is the purpose of the envelope $\mathcal{E}$. It contains the concatenation of k encrypted with each member's public key (using e.g. RSA). Upon reception of a ciphered message $(\mathcal{E}, \mathcal{C})$, a peer attempts to decrypt each portion of the envelope with its private key, until it succeeds (and gets k to decrypt $\mathcal{C}$) or fails. As it is required to encode descriptors in an onion cell to be able to deliver them to far relays without revealing the participant's identity, the final control cell is generated recursively, aggregating descriptors on each layer, as follow: $\mathcal{C}^i \leftarrow \mathrm{BE}([pk(R_1^i) \cdots pk(R_n^i)], (D^i, \mathcal{C}^{i+1}))$, where $\mathrm{BE}([pk...], \mathcal{M})$ is the broadcast encrypt function and $pk(R)$ the public key of relay R.

Sender-receiver anonymity. A `MOC` must be built in collaboration by the two parties while preserving the sender-receiver anonymity. To connect their circuits, users must agree on circuit identifiers, symmetric keys, and a common hop forming a service descriptor: $([I^1 \cdots I^m], [K^1 \cdots K^m], [R_1^c \cdots R_n^c])$. The service descriptor is generated by the sender, then published to the relay directory, and retrieved by the receiver. Each participant configures its part of the circuit, choosing his own relays: the sender configures hops from 0 to c and the receiver from c to m. Once configured by each party, a data cell can be sent end-to-end by crafting a relay cell recursively with K and I as follow: $\mathcal{C}^i \leftarrow \mathrm{SE}(K^i, (I^i, \mathcal{C}^{i+1}))$ where $\mathrm{SE}(k, \mathcal{M})$ is a symmetrical-key encryption algorithm (e.g. AES).

Interacting users must exchange a public key from a key pair generated for the occasion out of band to find and authenticate each other through the service directory. The sender's public key will be used to advertise and find the sender, whereas the receiver will be authenticated by its public key. To preserve confidentiality and anonymity, the service descriptor is encrypted with the receiver's public key. Such a scheme is similar to authenticated onion services over Tor.

Garbage collection. `MOC` are also garbage collected. Contrary to Tor, SAFE circuits are not bound to a connection, preventing tearing down a whole circuit while a relay goes offline. Instead, we bound circuits to timers: after not receiving any data for a certain amount of time, circuits are removed from circuit tables, and their associated connections are closed.

So far we have introduced how multipath circuits are managed, assuming that participants know how to generate circuits identifiers I, keys K and selecting relays R_j^i. However, while I and K are randomly generated, picking relays follow a more complex logic depicted in the following section.

4.3.4 Selecting Relays for Multipath Circuits

Circuits are composed of random relays chosen by the extremities of the circuit (participants). They are discovered by querying the relay directory that returns a random sample of the network. Safe uses an algorithm that packs n relays per-hop while ensuring that hop availability probability $P(H)$ remains above a certain threshold ϵ: $P(H) \geq 1 - \epsilon$.

Hop availability probability is time-dependent, thus computed at time t and 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 building a hop, we start by picking n random relays then compute the availability of the hop. If the availability is below the defined threshold (defined through ϵ), the lowest availability relay of the hop 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 it can influence network security. The worst-case being all relays having the same availability. The rationale is a high availability relay will not compensate for the presence of low availability ones. 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 all relays have the same availability $P(R)$ and find the minimum availability to meet n and ϵ criteria, thus we seek $P(R)$ for $P(H) = 1 - \epsilon$, results are described in equation 4.2.

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

So, 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 thresholded, 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 their hop, 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 amongst circuits to protect themself from the predecessor attack. 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 directory must then announce two predictions for each relay: $R^{[t,t+h_g]}$ and $R^{[t,t+h_s]}$.

Computation of relays availability probabilities. As relays are in charge of regularly registering themselves to the relay directory, the latter can log relays period of availability and unavailability. Based on these metadata, the directory can use predictors to compute the future of relays based on their past. Such information is used by the Tor relay directory to assign the "guard" flag to relays that, among other things, feature a high availability according to Tor internal rules, and that is expected to perform similarly in the future. Instead of announcing a flag to inform clients if, whether or not the relay can be taken as a guard, SAFE relay directory directly publishes its prediction results for the two considered horizons. It is then up to the client to decide when building a hop following the algorithm introduced previously if the relay is suitable or not. The choice of the predictor to compute the expected availability from the past uptime of a relay is let free to the relay directory implementer, and must probably be adapted to the situation. In the evaluation, we demonstrate the effectiveness of two simple predictors an EWMA and a Markov chain on our edge devices dataset.

As we have described our new onion routing protocol, in the following, we explain how we can leverage it to achieve a Safe and Anonymous File Exchange between users.

4.3.5 Leveraging Multipath Circuits: a File Exchange Application

With `MOC` a path taken by relay cells may change at any time due to relay congestion and availability. Data could then arrive out of order at the application layer, or can even be lost if a relay crashed, breaking any stream abstractions, as done by Tor. Instead of enforcing a stream abstraction at the link level, we deliver datagrams to the applications, similarly to UDP. Accordingly, SAFE is built on top of an anonymous UDP-like channel:

order and delivery of the messages are not guaranteed by the protocol. These features must be supplied by the application layer, in our case the file transfer protocol.

To build the Safe file exchange protocol, we implement the Selective Repeat Automatic Repeat-reQuest (ARQ) [141, 227, 167] algorithm, a sliding-window protocol that lets the sender send several chunks at once, and allows the receiver to accept them out of order. Each chunk is associated with its position (or ID). The sender sends the chunk and its ID in an onion cell on the `MOC`, while the receiver sends back an acknowledgment (ACK) with the same ID for each received piece in a relay cell again, on the same `MOC`. When the sender does not receive an ACK after sending a chunk, it retries sending after a timeout of several seconds. The file exchange completes once each file chunk has been ACKed.

4.4 Security Analysis

4.4.1 Attacks

In this section, we review common onion routing attacks and how they apply to Safe. We also put them in perspective with Tor to provide a comparison point.

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

Epistemic Attacks With Safe, we aim to drastically increase the number of relays hence triggering scalability issues to the directory server. It could be tempting to leverage edge devices too or to propose an ad-hoc solution, but the risk is very high to enable epistemic attacks discussed in Section 2.4. As we consider this problem as orthogonal as ours, real-world deployment of Safe would leverage one of the network discovery solutions discussed in Section 2.2.1 to fulfill its needs.

Users Exposition As we seek to make users contribute with their personal equipment, published relay information raises new questions on possibilities to infer users' behavior. Safe's design does not involve collecting more relay information than Tor, mainly relay

IP address and availability every hour. Incidentally, we built our dataset in Section 5.4 by collecting such data.

In contrast with Tor, however, the availability of edge devices is much more likely to be correlated with users' behavior than servers running in datacenters. The question is not simply about data collection because, as long as relay IP addresses are being advertised, many data can be inferred, such as user's sleep patterns, by simply querying the relay directory. If an attacker learns someone's IP address, they can infer whether or not it is a Safe user 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 2.4, 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 Safe 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.

Forward Secrecy Our presented protocol, on its own, does not provide forward secrecy. This means that an attacker recording traffic at one instant in time and later compromise the server's keys will be able to decipher the communication. The simplest approach to provide forward secrecy is to have relays rotating their keys regularly and republishing them to the directory server. While not appealing in the Tor case where the full consensus should be redownloaded way more often by clients, we already need for our design new directory models that do not serve the full consensus, like ConsenSGX [190] or PIR-Tor [148], making this approach perfectly conceivable.

We could also adapt the Tor Authentication Protocol (TAP) [58] to our design at the cost of an additional message exchange per configured relay. Under the hood, the TAP protocol performs a recursive encrypted Diffie Hellman key exchange with each hop of the circuit. Diffie Hellman enables (only) two parties (the client and the relay) to agree on a common secret. In our case, more share a common secret, as we consider multiple relays instead of one. To circumvent this limitation, a client would negotiate an independent secret with all relays of a hop then generate a new, common secret, and send it to all the relays of the hop. This scheme would require some modifications to our protocol, preventing the client to recursively encode the packet until the recipient: instead, similarly to Tor, the packet would be recursively decoded on the first circuit then recursively encoded on the second one.

TAP is considered a heavy protocol to construct onion circuits. Implementing our proposition would impact performance even more. The first proposal to simplify circuit construction, the ØS Protocol [161], introduced a way to open a circuit in one RTT but suffered from an attack [78] breaking its forward secrecy. `ntor` [78] has been proposed as a fix, which is slightly more computing-intensive than the original proposal. Finally, ACE [12], an optimization of `ntor`, reduced the computing complexity of opening a circuit. More ambitious approaches, like Sphinx [117], could also help simplify onion circuit construction while providing forward secrecy.

As a conclusion, we argue that a straightforward implementation of Safe would simply involve rotating keys periodically to provide eventual forward secrecy. We defer to a future work an adaptation of our protocol to support perfect forward secrecy in a lightweight and robust manner.

4.4.2 E2E Anonymity Quantification

Our goal is now to estimate what anonymity can a user expects from our security goals and the studied existing attacks. Especially, we want 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 our *Relationship Anonymity* security feature, we refer to this event as D_1. Attacker $\mathcal{A}$ would need to corrupt both receiver and sender entry guard, 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 Safe, 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 deanonymize a user, thus picking one attacker relay is enough in the entry hop is enough to conduct all attacks. The probability of breaking *Relationship Anonymity* with SAFE is then presented in equation 4.4.

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

An attacker $\mathcal{A}$ might only be interested in knowing if a targeted user sent a file in the network whatever 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.

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

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

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

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

If we consider same-size networks, then α will be lower than one and Tor will be more advantageous. If we seek the equilibrium, the SAFE 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 SAFE considered network is even bigger than n times the Tor one, then SAFE provides better anonymity probability against de-anonymization attacks studied.

Still, before comparing both networks, we must estimate the difference between the *considered* networks by the guard selection algorithm and the *whole* network as known by the relay directory. Indeed, in practice, both Tor and SAFE 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 SAFE, 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 taken 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.

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

We argue that in practice we could easily obtain a positive α considering the current Tor user/relay base. Currently, Tor features around 3000 guards over its 6000 available relays at a given point in time. For SAFE, 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 4.2), and keep two SAFE 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 a SAFE 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 user will run a relay, we argue that the ratio remains positive, 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 SAFE relay, it is possible to provide anonymity similar to Tor with SAFE's design. Passed this value, it is even preferable to use SAFE instead of Tor.

4.5 Evaluation

To evaluate Safe effectiveness against realistic data, we leverage the dataset introduced in Section 4.2 to simulate volatility. We then use this dataset to highlight Safe main strengths: high availability and throughput despite churn. In the last part, we discuss Safe 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 Safe 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 4.2. Excluding our microbenchmark, all our other results are based on this dataset.

4.5.2 Performance & Comparison

Safe fights churn both at the macro and micro levels. At the macro level, Safe 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, Safe 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 Safe 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 Safe 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 Safe 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%).

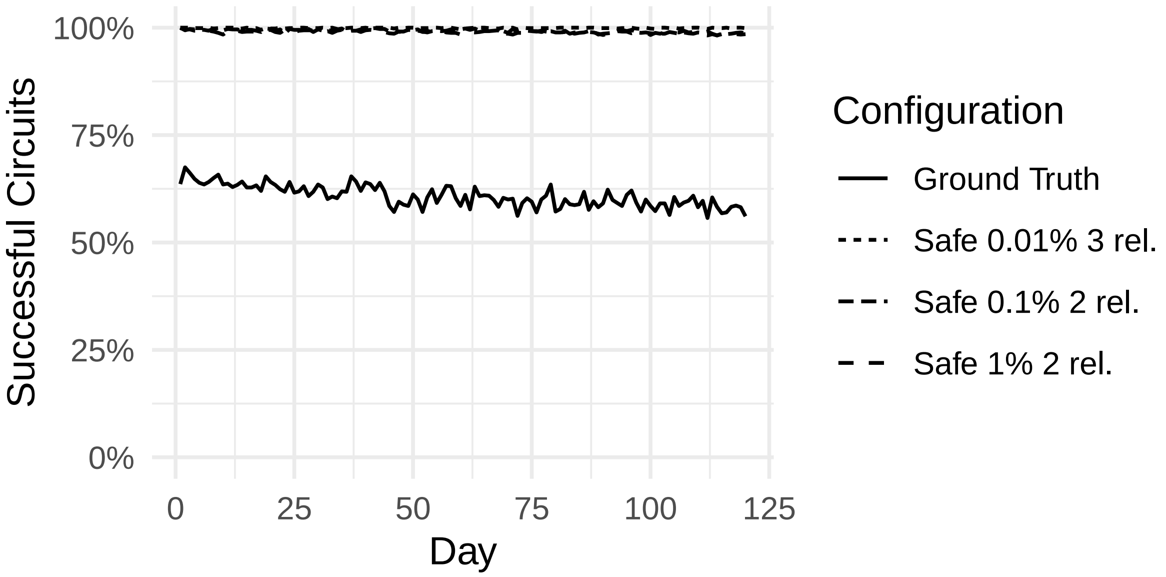


Figure 4.4 – During a guard lifetime, average ratio of working circuits for 1000 users.

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 Safe 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 Safe can build working multipath circuits across edge devices, we show that Safe can reliably and efficiently route onion cells through these circuits. To effectively evaluate our whole system, including our onion protocol, 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.

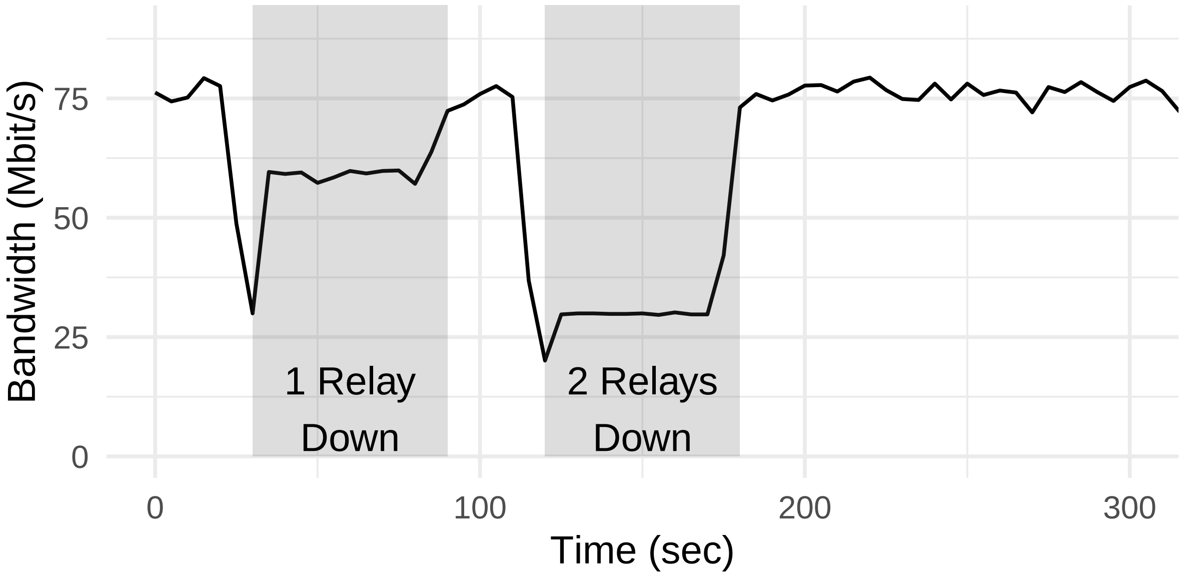


Figure 4.5 – Transfer bandwidth as experienced by end users over Safe with 30 Mbit/s relays

First, we see that despite two unavailability events, Safe 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 Safe 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 Safe 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

Safe 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.

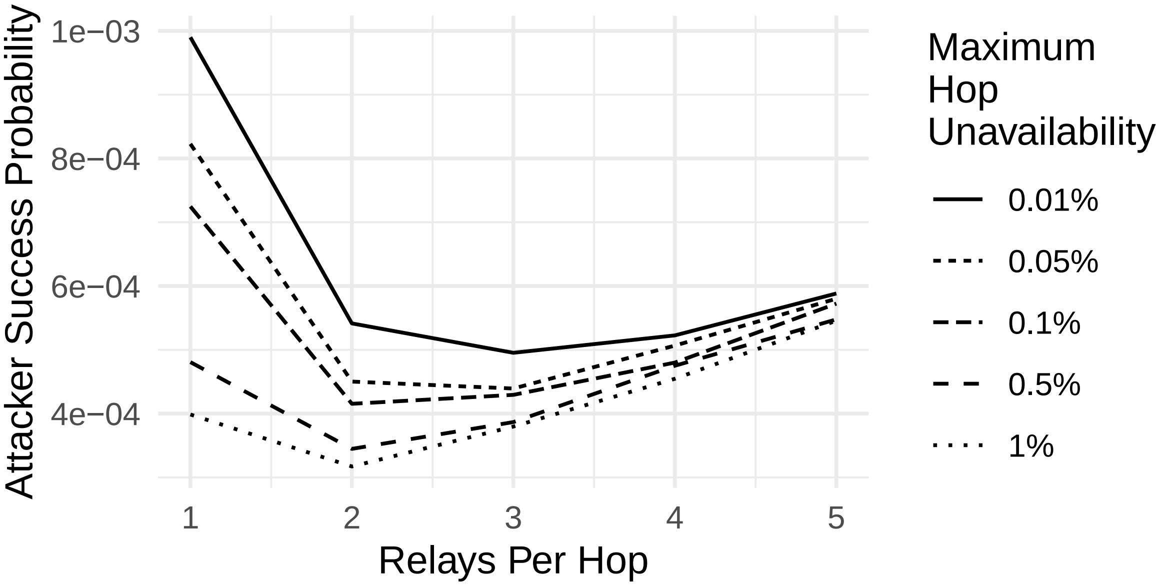


Figure 4.6 – Influence of the number of relays per hop on the anonymity quantification for a fixed availability target.

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.

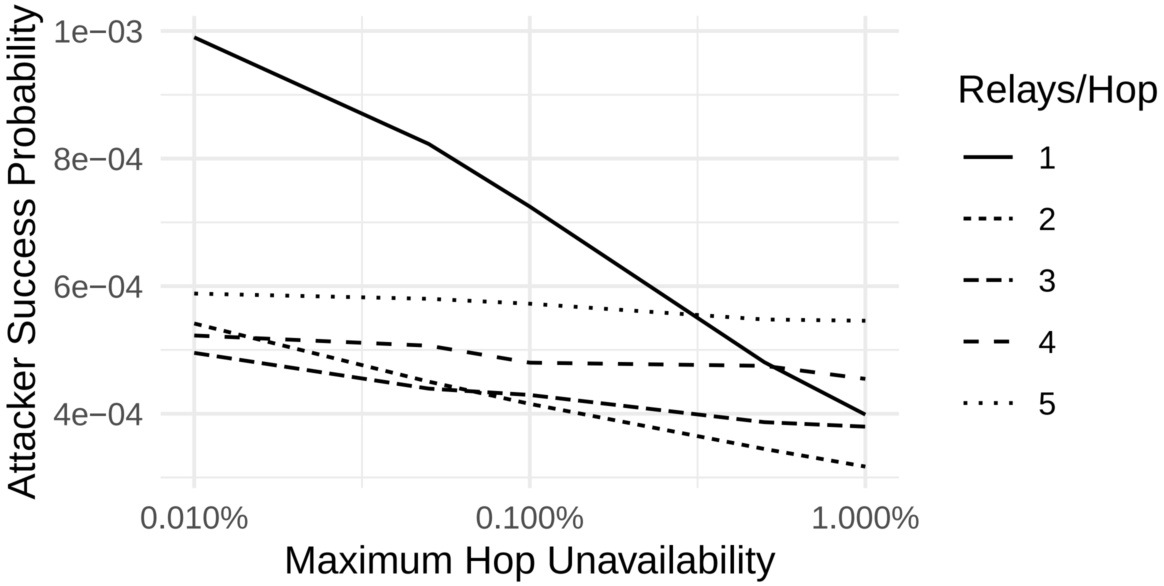


Figure 4.7 – Influence of the targeted availability ϵ on the anonymity quantification for a fixed number of hop.

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.

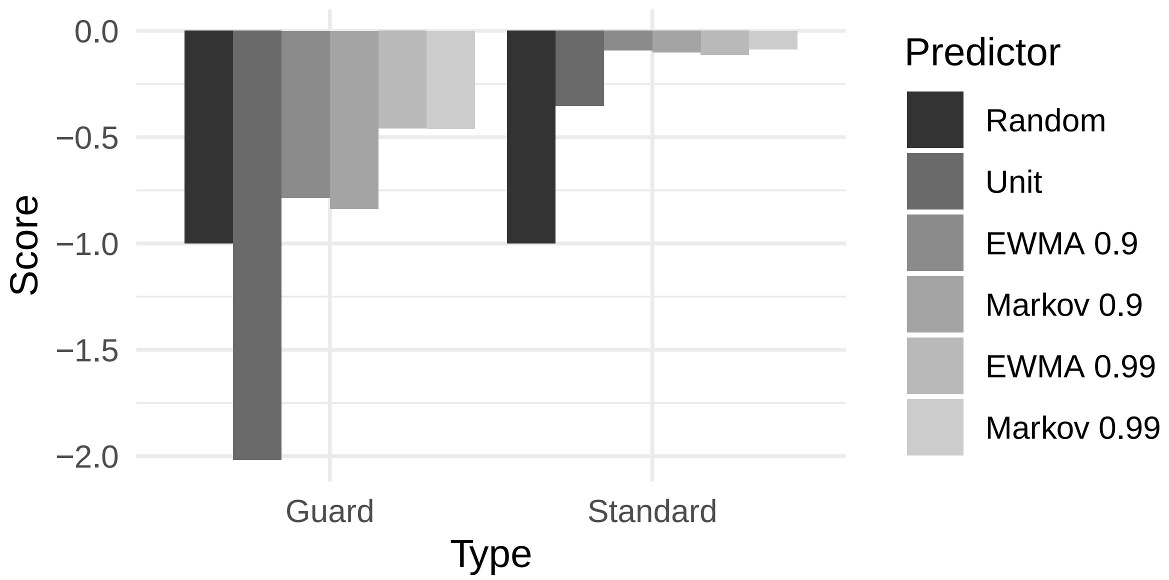


Figure 4.8 – Evaluation of the effectiveness of 3 predictors: random, EWMA and markov chain. We use a logarithm scoring rules, close to zero is better. Guard and Standard differs from their prediction horizon: 4 months and 1 hours respectively.

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

4.6 Conclusion

We presented Safe, a new onion routing anonymity network providing highly available, high throughput, and decentralized circuits by leveraging edge devices. Until yet, edge relays were discouraged in anonymity networks, including Tor, due to their volatility that lead to high circuit breakage. Considering that users at the edge are way more numerous than relays in the core (300× in Tor), we see high potentials in leveraging them to enable new usage that are impossible yet including large files transfer due to lack of bandwidth.

To build anonymous, available, and fast circuits overcoming volatility, we introduced a new design referred to as Multipath Onion Circuit (MOC). MOC uses multiple relays per-hop and thus can forward data as long as one relay per hop is still available. Combined with relay prediction, Safe is able to build MOC with availability close to 100%. Inside a MOC, Safe can efficiently route data with high throughput, achieving an effective 75 Mbit/s over 30Mbit/s relays by load balancing data through available relays in the MOC during a file transfer while dynamically adapting to faults in order to make them invisible to the end-user.

As Safe would enable the use of orders of magnitude more relays, we show that as long as 1% of the users run a relay, the success probability of a de-anonymization attack would be inferior to Tor despite the fact we pick more relays per circuit.

We plan to submit this work to the USENIX Security 2021 conference.

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 other ones 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 servers that can collect data or metadata (even with end-to-end encryption activated) would reduce privacy by enabling operators to build knowledge on their user's behaviour. 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 their 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 [108, 24], membership and failure detection [127, 55, 47, 52], aggregating data [109], and live streaming [70], [71]. This wide adoption comes from the resilience, simplicity, and natural distribution of gossip protocols [10, 127, 156]. These properties are particularly adapted to community-based networks where nodes are volatile and low powered. Furthermore, they do not require third party infrastructures.

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 [122][63][65].

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 protocol completeness, by sending linear combinations of several messages instead of a given plain message, which increases the probability of propagating something new to recipients [54, 85].

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[32], but this kind of techniques are not applicable to gossip protocols as they 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 its 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 [65], while keeping the same properties.

5.1 Network Coding Background

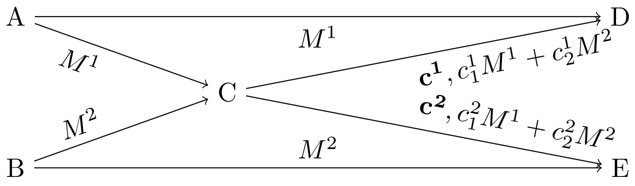

A
D
M^1
M^1
C
$\mathbf{c^1}, c_1^1 M^1 + c_2^1 M^2$
M^2
$\mathbf{c^2}, c_1^2 M^1 + c_2^2 M^2$
M^2
B
E

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

RLNC[68] 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 $c'_i = \sum_{j=1}^{m} d_j c^j_i$.

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 the messages 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 value must be assigned in such a way that enough messages are in the same generation to benefit from 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 [110]. 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 in 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 [128] 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 clue of 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 network coding, the number of needed packets corresponds to the number of missing messages. With network coding, 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 identifiers of 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).

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

Algorithm 1 Process RLNC Packets

```
$g \leftarrow 0$                                    ▷ Encoding generation
$rcv \leftarrow$ SET()                              ▷ Received packets
$ids \leftarrow$ ORDEREDSET()                       ▷ Known message identifiers
$dlv \leftarrow$ ORDEREDSET()                       ▷ Delivered message identifiers

function PROCESSPACKET(p)
    if $p = \varnothing$ then
        return False

    $\langle g^1, c^1, \_ \rangle \leftarrow p$
    $oldRank \leftarrow \textsc{Rank}(g^1, rcv)$
    $rcv \leftarrow \textsc{Solve}(g^1, rcv \cup \{p\})$

    if $oldRank = \textsc{Rank}(g^1, rcv)$ then
        return False                                ▷ Packet was useless

    for all $\langle id, \_ \rangle \in c^1$ do
        $ids \leftarrow ids \cup \{\langle g^1, id \rangle\}$      ▷ Register new identifiers

    for all $\langle g^2, c^2, e \rangle \in rcv$ do
        $\langle id, \_ \rangle \leftarrow c^2[0]$
        if $g^1 = g^2 \wedge \textsc{len}(c^2) = 1 \wedge \langle g^2, id \rangle \notin dlv$ then
            $dlv \leftarrow dlv \cup \{\langle g^2, id \rangle\}$
            DELIVER($e$)                            ▷ New decoded message

    if $g^1 > g \vee \textsc{Rank}(g, rcv) \geq 1$ then
        $g \leftarrow \textsc{max}(g^1, g) + 1$     ▷ Update Lamport Clock
    return True                                     ▷ Packet was useful
```

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 [65]. 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 which consists of exchanging recent message histories via a trading window. The trading window is generated by the GetHeaders function, and 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

Algorithm 2 Push

```
$k, ittl \leftarrow ...$                                  ▷ Push fanout and initial TTL
$dottl, dodie \leftarrow ...$                             ▷ Push strategy

function SendToNeighbourgs($h, headers$)
    for $k$ times do
        $p \leftarrow$ Recode($h, rcv$)
        Send(Push, $p, headers$)

function Broadcast($m, headers$)
    $id \leftarrow$ UniqueID()
    $p \leftarrow \langle g, \{\langle id, 1\rangle\}, m\rangle$
    $dlv \leftarrow dlv \cup \{\langle g, id\rangle\}$
    ProcessPacket($p$)
    $headers.ttl \leftarrow ittl$
    SendToNeighbourgs($g, headers$)

function NCPush($p, headers$)
    $\langle h, \_, \_\rangle \leftarrow p$
    if ProcessPacket($p$) $\vee \neg dodie$ then
        if $dottl \wedge headers.ttl \leq 0$ then
            return
        if $dottl$ then
            $headers.ttl \leftarrow headers.ttl - 1$
        SendToNeighbourgs($h, headers$)
```

Algorithm 3 Pull

```
$rotation \leftarrow 0$                                    ▷ Rotation position

function NCPullThread($headers$)
    $ask \leftarrow$ OrderedSet()
    $rotation \leftarrow rotation + 1 \mod |ids \setminus dlv|$
    for all $m \in$ Rotate($ids \setminus dlv, rotation$) do
        $ask \leftarrow ask \cup$ Gen($m, rcv$)
    Send(Pull, $ask, headers$)

function NCPull($asked, headers$)
    $p \leftarrow \varnothing$
    if $\exists\, g \in asked,$ Rank($g, rcv$) $> 0$ then
        $p \leftarrow$ Recode($g, rcv$)
    Send(PullReply, $p, headers$)

function NCPullReply($p$)
    ProcessPacket($p$)
```

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

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

Algorithm 4 CHEPIN-Pulp

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

function GetHeaders
    $start \leftarrow \textsc{max}(0, |ids| - sm - tm)$
    $end \leftarrow \textsc{max}(0, |ids| - sm)$
    return $\{tw : ids[start : end]\}$

upon receive Push($p, headers$)
    $ids \leftarrow ids \cup headers.tw$
    NCPush($p$, GetHeaders())
upon receive Pull($asked, headers$)
    $ids \leftarrow ids \cup headers.tw$
    NCPull($asked$, GetHeaders())
upon receive PullReply($p, headers$)
    $ids \leftarrow ids \cup headers.tw$
    NCPullReply($p$)

thread every $\Delta_{pull}$
    NCPullThread(GetHeaders())
thread every $\Delta_{adjust}$
    $missingSize \leftarrow |ids| - |rcv|$
    if $missingSize > prevMissingSize$ then
        $\Delta_{pull} = \frac{\Delta_{adjust}}{missingSize - prevMissingSize + prev_{useful}}$
    else if $missingSize > 0 \wedge prev_{useless} \leq prev_{useful}$ then
        $\Delta_{pull} \leftarrow \Delta_{pull} \times 0.9$
    else
        $\Delta_{pull} \leftarrow \Delta_{pull} \times 1.1$
    $\Delta_{pull} \leftarrow \textsc{max}(\Delta_{pull}, \Delta_{pull_{min}})$
    $\Delta_{pull} \leftarrow \textsc{min}(\Delta_{pull}, \Delta_{pull_{max}})$
    $prev_{useless}, prev_{useful} \leftarrow 0$
    $prevMissingSize \leftarrow missingSize$
```

5.4.1 Experimental setup

Our experiments consist in disseminating 1 000 messages at a rate of 150 messages per second, each message being emitted by a different nodes 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 is run 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[237]. Latency averages at 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.

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.

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

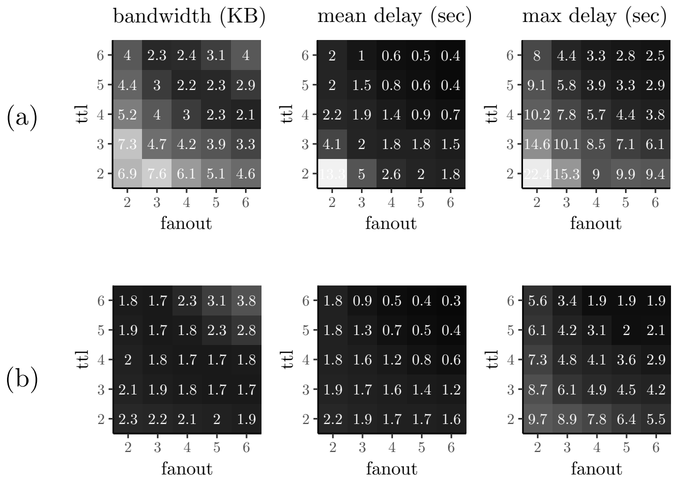


Figure 5.2 – Pulp (5.2a) and CHEPIN-Pulp (5.2b) behavior under various configuration of the protocol (fanout and time to live)

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 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

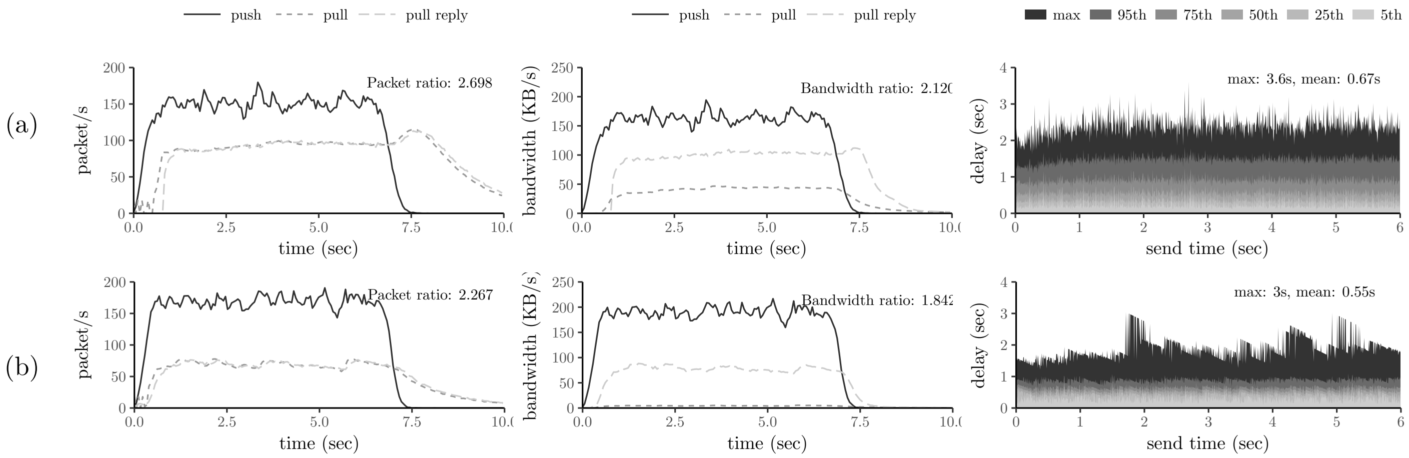


Figure 5.3 – Comparison of exchanged packet rate, used bandwidth and message delay for 5.3a Pulp and 5.3b CHEPIN-Pulp

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 seeing 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

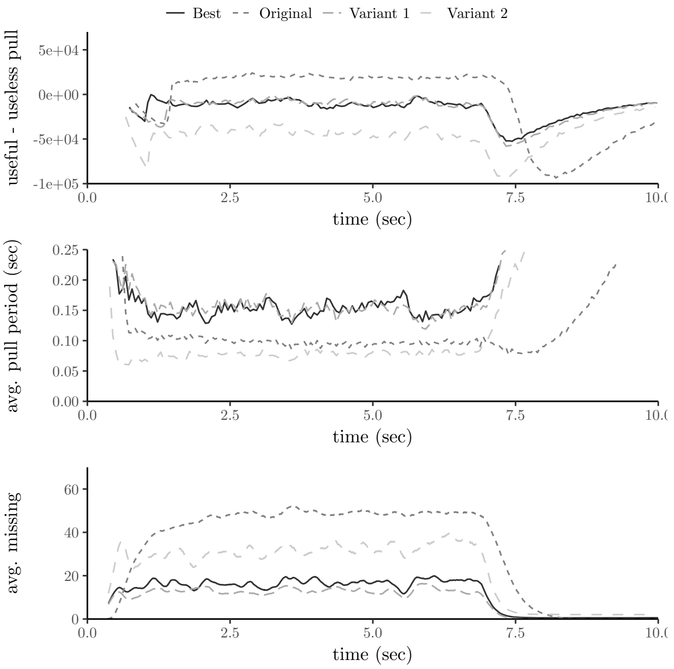


Figure 5.4 – How adaptiveness algorithms impact the protocol efficiency

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: 2.4 instead of 1.83 for our solution. This shows the importance of the modifications we made to the Pulp protocol.

On 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

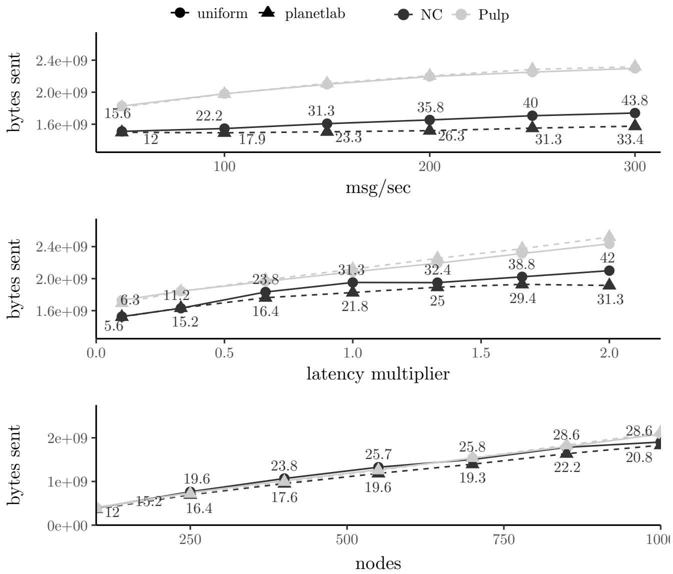


Figure 5.5 – Variation of network configuration. The number near each point indicates the average generation size for NC.

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 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[20]. 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.

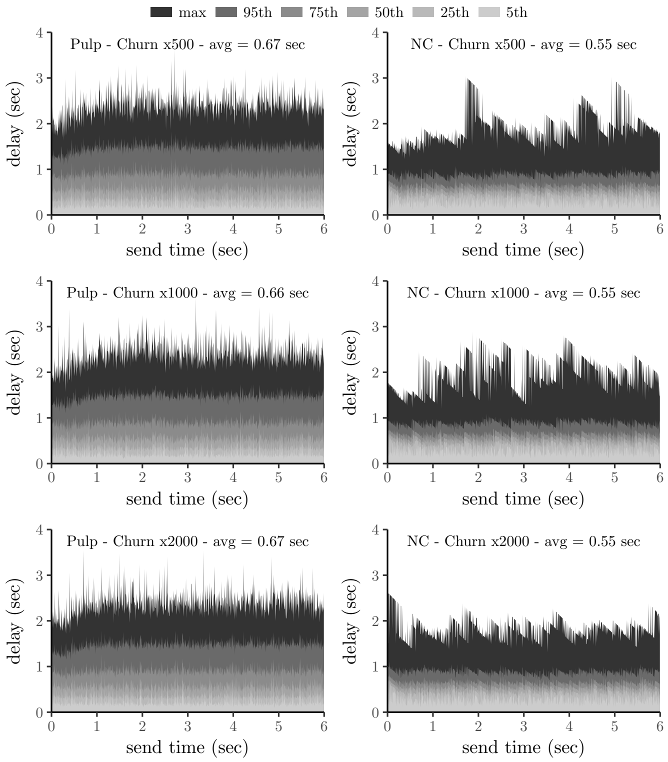


Figure 5.6 – Delay observation with churn on the system

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 members 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 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 extend 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 [25].

In the following, we analyze the synergies that could emerge from our three contributions Donar, Safe 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, Safe 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 peering to WiFi including the "last mile" link (ADSL, coaxial, etc). In some cases, our current algorithm directly address these problems. For example, thanks to its probing, Donar will take the path having the less 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, Safe focuses on organic scaling on a standalone implementation: the two systems are very different and we think that synergies exist between Safe 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 in Safe'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. In this case, the protocol must be able to resist malicious behaviors, we note especially Sybil attacks [61], where the attacker floods the entire network with its profiles and Eclipse attacks [197], where the attacker circles a target with its profile. In this context, a byzantine-resistant RPS [23] 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 requires to 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 on 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 lead 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 identity as she wants to separate different part of their life. Communication are 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 Services primitives, identifying patterns, proposing building blocks on top of Onion Services 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 amount of anonymity networks were designed, only a more restrained set was effectively deployed, and none of them support the wide variety of today communication. To make 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 networks specificities in mind and evaluated in real conditions. As a result, we were able to make real calls while meeting industry standards in term of quality of service.

Safe 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 Safe, it is no longer the case as our contribution captures the notion of availability at its heart, provision redundancy accordingly and seamlessly adapt 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 property is given by gossip protocols, that naturally embed redundancy but require to send more messages. Introducing network coding enables 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.

Bibliography

[1] *#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).

[2] 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).

[3] 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.

[4] 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.

[5] 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.

[6] Mashael AlSabah and Ian Goldberg, « Performance and security improvements for tor: A survey », *in*: *ACM Computing Surveys (CSUR)* *49.2* (2016), pp. 1–36.

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

[8] 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.

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

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

[11] 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.

[12] 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.

[13] 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.

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

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

[16] 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.

[17] 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).

[18] 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.

[19] 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).

[20] 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.

[21] 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.

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

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

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

[25] 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.

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

[27] 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).

[28] 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.

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

[30] 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).

[31] 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.

[32] 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.

[33] 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.

[34] 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.

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

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

[37] 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.

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

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

[40] *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).

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

[42] 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.

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

[44] 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.

[45] 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).

[46] 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).

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

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

[49] 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.

[50] 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.

[51] Norman Danner et al., « Effectiveness and detection of denial-of-service attacks in Tor », *in*: *ACM Transactions on Information and System Security (TISSEC)* (2012).

[52] 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.

[53] 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.

[54] 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.

[55] 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.

[56] 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.

[57] 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.

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

[59] 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).

[60] 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.

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

[62] 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.

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

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

[65] 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.

[66] Amos Fiat and Moni Naor, « Broadcast Encryption », en, *in*: *Advances in Cryptology — CRYPTO' 93*, ed. by Douglas R. Stinson, Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1993, pp. 480–491, ISBN: 978-3-540-48329-8.

[67] 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.

[68] 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.

[69] 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.

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

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

[72] 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.

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

[74] 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.

[75] 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.

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

[77] 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.

[78] 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.

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

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

[81] 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.

[82] *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).

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

[84] 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`.

[85] 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.

[86] 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.

[87] 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.

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

[89] 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).

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

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

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

[93] 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).

[94] *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).

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

[96] 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.

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

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

[99] 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`.

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

[101] 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.

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

[103] 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.

[104] Rob Jansen and Matthew Traudt, « Tor's been KIST: A case study of transitioning Tor research to practice », *in*: *arXiv preprint arXiv:1709.01044* (2017).

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

[106] 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.

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

[108] 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.

[109] 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.

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

[111] 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.

[112] 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.

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

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

[115] 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.

[116] 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.

[117] Aniket Kate and Ian Goldberg, « Using sphinx to improve onion routing circuit construction », *in*: *International Conference on Financial Cryptography and Data Security*, Springer, 2010, pp. 359–366.

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

[119] 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.

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

[121] 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.

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

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

[124] 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.

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

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

[127] 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.

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

[129] 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.

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

[131] 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.

[132] 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.

[133] 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.

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

[135] *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).

[136] 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).

[137] 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.

[138] 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).

[139] 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).

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

[141] 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.

[142] 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).

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

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

[145] 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).

[146] 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.

[147] 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.

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

[149] 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.

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

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

[152] 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.

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

[154] 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.

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

[156] 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.

[157] 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.

[158] *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).

[159] 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.

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

[161] 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.

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

[163] 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.

[164] 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).

[165] 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).

[166] 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).

[167] 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.

[168] 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.

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

[170] 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).

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

[172] 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).

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

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

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

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

[177] 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.

[178] 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`.

[179] 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.

[180] 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.

[181] 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.

[182] 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).

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

[184] 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`.

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

[186] 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).

[187] 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).

[188] 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).

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

[190] 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`.

[191] 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.

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

[193] 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.

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

[195] 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.

[196] 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.

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

[198] 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.

[199] 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.

[200] 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).

[201] Douglas R Stinson, *Cryptography: theory and practice*, Chapman and Hall/CRC, 2005.

[202] 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).

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

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

[205] 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.

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

[207] 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.

[208] 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).

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

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

[211] 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).

[212] 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.

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

[214] *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).

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

[216] 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.

[217] 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.

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

[219] *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).

[220] 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).

[221] 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.

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

[223] 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.

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

[225] 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).

[226] 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).

[227] 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`.

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

[229] *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).

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

[231] 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.

[232] 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.

[233] 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.

[234] 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).

[235] 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.

[236] 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).

[237] 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.

[238] 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).

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 appelées 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. Safe 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. Safe 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.