

École polytechnique de Louvain

Topology Inference in GossipSub: Passive and Active Reconstruction via Peer Exchange Exploitation

Author: **Virgile DEVOLDER**

Supervisor: **Etienne RIVIERE**

Readers: **Matthieu PIGAGLIO, Peter VAN ROY, Etienne RIVIERE**

Academic year 2025–2026

Master [120] in cybersecurity with focus Cryptanalysis and Forensics

Abstract

Peer-to-peer networks form the foundation of critical decentralized infrastructures such as Ethereum and IPFS, whose operation relies on the efficient propagation of information via GossipSub, a popular publish-subscribe protocol. The topology of GossipSub, the logical network between machines that guides message transmission, has a direct impact on the network’s performance and, above all, its security.

This thesis investigates the extent to which it is possible to reconstruct the GossipSub topology through passive and active observation, considering adversarial models ranging from a single passive node to a coordinated coalition. Three inference methods are implemented in an extended Java simulator, whose topological fidelity is validated against a real libp2p/GossipSub deployment on Grid5000: passive inference of the local neighborhood, passive coalition inference, and active inference exploiting the peer exchange mechanism.

Experiments on networks of 1,000 nodes show that a single passive observer reconstructs its local neighborhood with perfect precision in fewer than three heartbeats, and a coalition of eight nodes achieves up to 6% global node coverage, with coverage scaling linearly in the coalition size. Inference leveraging peer exchange extends this reach significantly: eight nodes discover up to 31% of the network in a vulnerable scenario, and up to 93% with deeper inference. In every case, however, the nodes themselves are identified but the links between them are reconstructed with low precision: the inference reveals who is in the network, not the exact shape of the topology. Discovered nodes are recovered with 100% precision, though at the cost of up to 33% topology drift at depth two. These figures, however, assume that peer exchange is enabled network-wide, a deliberately permissive setting: in realistic deployments PX is restricted to a few bootstrap or trusted nodes, which substantially narrows the attack surface. Appropriate degree tuning and PX-side defenses such as extended backoff or suspicious GRAFT detection greatly limit inference effectiveness. These results show that GossipSub’s protocol signals constitute an information channel about network structure, and that partial topological knowledge is sufficient to prepare targeted attacks such as eclipse or de-anonymization attacks.

Acknowledgements

I would like to thank my supervisor, Professor Etienne Riviere, for his support, availability, and invaluable guidance throughout the year, as well as Matthieu Pigaglio and Professor Peter Van Roy for agreeing to serve as readers for this thesis. I am also grateful to EPL for providing a server on which to run the necessary tests, and to Grid5000, a collaborative university server-sharing network, which proved invaluable in validating the results. Finally, my deepest thanks go to my family, friends, and girlfriend for their daily support.

Preface

During the writing of this thesis, AI was used for various purposes. The purpose of this section is to clarify what it was used for and what it was not used for.

DeepL was used for translation and to ensure proper phrasing in English. The AI was used solely for phrasing. Grammarly was used in certain places to correct and improve phrasing.

AI was also used to improve and assist in the development of the programs that were used to obtain the results of this thesis.

AI was not used to compile the bibliography, it was done entirely by hand. Furthermore, the content is entirely my own writing, as are the various ideas and analyses presented in this thesis.

Contents

Abstract	i
Acknowledgements	ii
Preface	iii
1 Introduction	1
1.1 Contributions	3
1.1.1 GossipSub	3
1.1.2 Methods for topology reconstruction	4
1.1.3 Simulator and experimental evaluation	4
1.1.4 Security implications	6
1.1.5 Defenses	7
2 Problem Statement	8
2.1 Problem Definition	8
2.2 Challenges in Topology Inference	9
2.3 From topology measurement to security implications	10
2.4 Security analysis of GossipSub	11
2.5 Adversarial Model	12
3 Background	13
3.1 Gossip-based dissemination protocols	13
3.2 Design and objectives	13
3.3 Overview of GossipSub	14
3.4 Core mechanisms of GossipSub	15
3.4.1 Topic-based Mesh Construction	16
3.4.2 Message propagation mechanisms	18
3.4.3 Peer scoring system	20
3.4.4 Heartbeat and mesh maintenance	22
3.5 Additional protocol mechanisms	23

3.5.1	Peer exchange (PX)	23
3.5.2	Peer discovery	25
4	Simulator	26
4.1	Design objectives	27
4.2	Simulation environment and performance	28
4.3	Architecture	29
4.4	Modeling of GossipSub	31
4.5	Ground-truth	33
4.6	Fidelity assessment	35
4.6.1	Metrics	35
4.6.2	Context	36
4.6.3	Analysis of the results	37
4.6.4	Conclusion	42
5	Experiments and Data analysis	45
5.1	Evaluation methodology	45
5.1.1	Scope of the inference methods	45
5.1.2	Scenarios	46
5.1.3	Sources of topological information leakage	47
5.1.4	Architecture of inference methods	48
5.1.5	Metrics	49
5.2	Results for passive ego-mesh inference	51
5.3	Results for passive coalition inference	55
5.4	Results for PX flood	59
5.4.1	Global coverage	59
5.4.2	Neighborhood Reconstruction by Target	61
5.4.3	Wave progression	63
5.4.4	Topology drift	65
5.4.5	Discovery bias	66
5.5	PX flood on the real GossipSub	68
5.6	Countermeasures against PX flood	69
5.6.1	Defense design	69
5.6.2	Effect on global coverage	70
5.6.3	Effect on attack progression	71
5.6.4	Limits	72
5.7	Discussions	74
6	Related Work	78
6.1	Early P2P networks and topology measurement	78
6.2	Modern large-scale P2P systems	79

6.3	Privacy and deanonymization	80
7	Conclusions	82
7.1	Future work	83

Chapter 1

Introduction

Over the last two decades, distributed systems have become central to modern computing. More than just a trend, they have become an alternative to many existing systems: financial systems, content distribution, and decentralized data infrastructures. These systems rely entirely on P2P (peer-to-peer) networks to function. Today, they represent a major economic and technological ecosystem opportunity with blockchains such as Bitcoin [1] or Ethereum [2]. These systems are popular because trust lies within the system itself, there is no central authority.

In these P2P systems, there is no global coordination: each node is autonomous and has access only to a partial and local view of the network. These networks are open to all and dynamic. Anyone can, if they wish, join a network and participate in it without requiring permission from a central authority. This lack of coordination is a key feature of decentralized systems and what makes these systems resilient, but it also introduces challenges in terms of observability and security [3].

Within these networks, the dissemination of information is critical. In the case of a blockchain, for example, new blocks and transactions must be propagated as quickly as possible across the network so that everyone has the same version of the blockchain. If information does not propagate quickly enough, two different states of a blockchain are created, this is known as a fork¹, and is what we try to avoid. The system's performance depends on the propagation latency and the effective coverage of the network. For all such networks, the dissemination protocol is a key component of a P2P system.

Pub-sub (publish-subscribe) protocols are employed to address these requirements. These protocols decouple producers from consumers and facilitate efficient information dissemination. Nevertheless, they encounter challenges related to scalability and robustness.

¹A fork occurs when two validators or miners produce conflicting blocks, resulting in two different chain states.

To address these challenges, GossipSub, proposed by Vyzovitis et al. [3], is a libp2p² protocol that has established itself as a widely adopted solution, particularly in systems such as Ethereum and IPFS [4]. Its principle is based on a hybrid approach: each node maintains a small set of direct neighbors, called a mesh, to which it transmits messages in their entirety, while periodically announcing the existence of these messages to a larger set of peers via metadata. These peers can then request the full content if they have not yet received it. This combination of structured broadcasting and probabilistic gossip allows GossipSub to strike a balance between performance and robustness, whereas naive broadcasting to all neighbors wastes too much bandwidth and pure gossip does not guarantee sufficient coverage.

In P2P networks, one of the key components is topology: the structure of the logical network used by the dissemination protocol. This topology is of interest because it influences latency, information redundancy, and propagation paths. Not all nodes are equivalent, some are central or strategic. However, the overall topology is not visible to anyone because all nodes have only a local view as shown in the Figure 1.1. As observed in older P2P networks, topology impacts performance and robustness [5], [6].

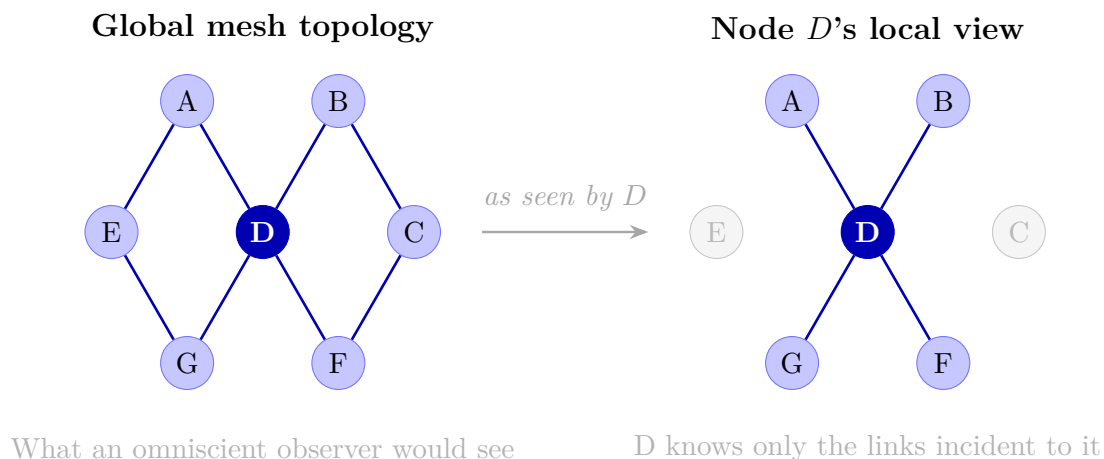


Figure 1.1: The same GossipSub mesh seen globally (left) and from the perspective of a single node (right).

Each node has a limited view, but interactions at the protocol level leave traces. Control messages, neighbor changes, and propagation patterns may leak information about the local network structure.

²libp2p is a peer-to-peer networking library that enables the creation of distributed applications capable of communicating with one another.

This raises a fundamental question: to what extent can the topology of a GossipSub network be inferred through passive observation or active participation? In practice, it is not that simple, there are several challenges to overcome. The network is dynamic, so one cannot rely on a snapshot in time.

Understanding the topology is a major security concern. Knowledge of the structure enables targeted attacks such as network partitioning and eclipse³ attacks [7], [8]. It also allows for deanonymization⁴ attacks [9]. Topology inference therefore poses a security risk.

Potential attacks against GossipSub are well known [3], but there has been limited research on topology reconstruction related to GossipSub. In particular, the extent to which GossipSub leaks structural information remains largely unexplored. This work will focus on this topic.

1.1 Contributions

This thesis contributes to the study of topology inference on GossipSub by examining the issue from a security perspective. Its contributions are as follows:

- A detailed analysis of how GossipSub works: a focus on topology and information leakage.
- The design of active and passive methods for reconstructing the topology of GossipSub.
- The extension of an existing GossipSub simulator written in Java to include additional features, as well as the validation of this simulator.
- An experimental evaluation of the reconstruction of the GossipSub topology under various conditions.
- A discussion of the security implications of topology inference in GossipSub and the risks enabled by partial topological knowledge.
- A proposal and evaluation of defenses against exploited active inference.

1.1.1 GossipSub

This thesis devotes an entire chapter to a detailed examination of the topological mechanisms underlying GossipSub. This description is based on an analysis of the paper that originated GossipSub [3], but also on an analysis of the Go code of

³Attack in which a node is isolated from the rest of the network by malicious peers.

⁴Attack in which the identity behind a node is revealed from its network behavior.

libp2p [10]. This analysis provides a better understanding of the inner workings of GossipSub and the reasons behind its various behaviours.

The analysis highlights that the topology is not a static graph, but rather a dynamic graph that evolves over time and is the result of the aggregation of local decisions made by each node. Mesh formation, scoring, pruning and opportunistic **GRAFT** are mechanisms that can influence the structure.

This knowledge also helps to highlight the various signals and mechanisms that provide useful information for inferring topology. This category includes signals such as **GRAFT** and **PRUNE**, as well as mechanisms less directly related to topology, such as **IHAVE** and **IWANT**⁵.

This GossipSub background chapter explaining topological mechanisms forms the theoretical foundation underpinning all this work.

1.1.2 Methods for topology reconstruction

One of the main contributions of this work is the development of various models for inferring the topology of GossipSub. These models have not been developed at random, rather, they form a series of building blocks, with each subsequent model building on the previous ones.

The first type of network observation is passive observation. Firstly, at a local level, a single observer attempts to reconstruct their own ego-mesh⁶ by interpreting the protocol signals they can observe passively. This model captures the observation surface available to a single node in the network. The second model is a passive model that extends the first: a coalition. The idea is to proceed as in the first model, but with several nodes combining their information to reconstruct a larger network. This demonstrates how topological knowledge can be accumulated without active participation.

The second type of observation is active observation. Observers no longer merely observe, but actively participate in the network to obtain more information. Active inference exploits a design flaw in the GossipSub peer exchange⁷ mechanism to extract information about network peers.

1.1.3 Simulator and experimental evaluation

An initial event-driven simulator [11] already existed and was based on PeerSim [12], a widely used Java simulator for scientific research into peer-to-peer networks. This

⁵The various control messages and the mechanisms described above are described in more detail later in the Chapter 3.

⁶The ego-mesh of a node refers to that node and its immediate neighbors within a topic.

⁷The GossipSub mechanism, which provides a list of peers to connect to, this mechanism is described in detail in the Chapter 3.

simulator had a branch dedicated to GossipSub. The simulator already implemented Kademlia⁸ alongside GossipSub for peer discovery. As for GossipSub, it was partially implemented: there was mesh and heartbeat logic, basic configurations, and an initial form of visualisation.

The starting point was a functional simulator, but one that was not fully instrumented to provide topological ground truth, nor did it feature topology inference simulations. Finally, it lacked campaigns, evaluation, and aggregations.

I have made a number of contributions on this GitHub⁹. The first is the transformation of the simulator into a topological measurement tool. Topological instrumentation for GossipSub has been added to enable the export of complete mesh snapshots, thereby producing a verifiable baseline for evaluation. This allows us to compare the inferred topologies with the actual simulated topology. We can also extract precise statistics and calculate values such as True Positive, False Positive and False Negative.

Next, the simulator’s topological fidelity relative to GossipSub was improved. Several key mechanisms governing the formation and evolution of the mesh have been added or improved to ensure experiments are conducted on a more accurate topology. These mechanisms include: local scoring, PRUNE backoff, peer exchange, opportunistic grafting, outbound quota constraints, and the management of retained or replaced peers. There are also mechanisms indirectly related to topology, such as the distinction between known, connectable, connected, reserve, and mesh member peers. These mechanisms are important because they alter the local stability of the mesh and modify the observable signals for someone seeking to infer the topology.

In addition to enhancing the simulator itself, a better experimental framework has been introduced. New scenarios have been added to account for various cases: the presence and absence of scoring, degraded peer pressure, the presence of PX, out-degree constraints, and reproducible seeds.

To utilise this experimental framework, various inference models have been developed. The first is a passive model. The initial building block was a single observer attempting to passively infer its immediate surroundings. Next, a coalition of several observers who passively infer their surroundings and pool their data to reconstruct a larger graph. To reconstruct the topology, they use messages they see passing by, such as GRAFT or PRUNE.

Following this passive model an active model has been developed. This model targets a node, joins its mesh so that it becomes overloaded, and causes the targeted node to suggest other nodes to connect to. Using this information, the process is repeated several times on the suggested nodes to expand the knowledge base.

To utilise the results of these models, a complete experimental pipeline has

⁸A P2P protocol that enables the efficient retrieval of data across a distributed network.

⁹<https://github.com/V1rg1lee/kademlia-simulator/tree/gossip-complete>

been implemented in Python. These Python scripts enable the launch of runs, the evaluation of each inference, multi-seed aggregation, the production of summary files of the results, and finally the generation of figures.

In order to evaluate the results correctly and put them into perspective, a measure of topology disruption was implemented to gauge the extent to which an active inference disrupts the topology and thus invalidates its own results. When an active observation is initiated, another run is launched, or the observers remain passive to see how the topology would have evolved without the active observation. The two final topologies are then compared, thereby measuring the disruption to the topology.

The results suggested that achieving global coverage is a difficult goal. However, depending on how GossipSub is configured, certain mechanisms may result in only partial knowledge of the overall topology.

The simulator was experimentally validated by comparing its results against a real deployment of GossipSub over libp2p. As the reference implementation, we reused the libp2p-based prototype from PANDAS [13], [14], which we adapted to serve as a minimal, standalone GossipSub deployment. The adaptation is available on this GitHub¹⁰.

1.1.4 Security implications

The following contribution of this work is a discussion of the security implications arising from the inference results. Topology inference is not an end in itself, the value of this inference lies in utilizing the information gained to subsequently launch an attack based on that information. Partial knowledge of the network, or of a part of the network, can make an attack possible or simply more effective.

This work therefore demonstrates that even an incomplete reconstruction can reveal important structural information. Depending on the inference, this may expose a node’s neighbors, observable boundaries, or a specific region. Based on this, one could launch an eclipse attack to slow down the propagation of information across part of the network, or make access to information impossible for a specific node.

Another implication is that changing certain protocol parameters can make inference more difficult and thus deter potential eavesdroppers within the network. In this work, several GossipSub configurations are tested to identify the best ones.

This thesis does not claim that GossipSub is insecure because part of the topology can be guessed, but demonstrates that the protocol exposes observable signals that allow security to be called into question depending on the resources deployed. The work establishes a link between what can be observed, reconstruction,

¹⁰<https://github.com/Virg1lee/GossipSub-Minimal>

and security risk.

1.1.5 Defenses

Finally, this work proposes a defense against the active inference described above. This defense is implemented in the simulator, and the inference results are compared with and without the defense. The goal of this defense is to slow down an inference as much as possible and to stop it completely if possible. The results of the proposed defenses show that active inference can be stopped in most cases, and significantly slowed down in the rest. This defense can be configured to suit the application using GossipSub. Crucially, it simply adds code to the GossipSub node, leaving everything else unchanged, making it very easy to deploy.

Chapter 2

Problem Statement

In a GossipSub overlay, the topology is not fully observable. Each node has a local view of the topology. The protocol handles broadcasting, which is a critical mechanism in P2P networks.

The topology influences latency, coverage, and the propagation of information. Not all nodes have the same importance in the network due to their advantageous or non-advantageous positions. Knowing the network topology allows us to understand this information.

In such a network, interactions between nodes leave marks. The way nodes behave provides clues. And certain messages convey information. By combining these sources of information, one can attempt to reconstruct the topology that was originally hidden. This raises the question of to what extent these observable signals are sufficient to reconstruct the topology.

With this reconstructed topology, one can determine the positions of the nodes and their connectivity with the rest of the network. Once this information is available, a targeted attack can be developed.

2.1 Problem Definition

A GossipSub network is a decentralized overlay network in which each node maintains only a partial, local view of the topology. Each node maintains its own mesh for a given topic and does not have access to the overall network structure. The overall topology is therefore hidden and is constantly defined by the aggregate of the local decisions made by the network's nodes.

An observer participating in the network can nevertheless access several forms of local evidence. It can observe protocol interactions exchanged with neighboring nodes, information related to message dissemination, local timing information, and its own direct connections. While these observations do not reveal the global

topology directly, they may still leak structural information about the hidden communication mesh.

The problem this thesis seeks to address is as follows: given only the protocol messages and local information, to what extent is it possible for an observer to reconstruct the overall topology of GossipSub? This applies whether the observer is acting alone, as part of a coalition of observers, or, more generally, within a portion of the network. In summary, the aim is to infer the relationships between network nodes from partial observations, and to determine to what extent the network can be reconstructed using passive and active observations.

Two observation modes are considered. In passive mode, the observer utilises only the information exposed to them whilst participating normally in the network. In active mode, the observer strategically places the nodes of its coalition in order to obtain topological information.

2.2 Challenges in Topology Inference

Inferring the topology of GossipSub is intrinsically difficult for several reasons.

First of all, observability is local. Each node can only see its immediate neighbourhood, its mesh connections and the protocol events in which it is involved. This means that the information available to an observer is fragmented across all the nodes in the network. No participant has access to the entire mesh, and if an observer makes an inference, no one can tell them whether their inference is correct because there is no oracle to confirm it.

Secondly, GossipSub is dynamic. The mesh is continuously maintained through heartbeat-driven decisions such as **GRAFT**, **PRUNE**, scoring-based retention, opportunistic grafting, and peer replacement. Even if an observer makes assumptions at a given moment, there is no guarantee that this assumption will still be correct later on. The topology cannot therefore be considered a fixed graph. Inference must therefore take into account the fact that the topology evolves.

Thirdly, the size of the network means it is unrealistic to expect to reconstruct the entire network. GossipSub is designed to support large decentralized systems in which nodes maintain their own local mesh. Even if an observer operates several nodes within the network, they cannot expect to observe the entire network. For this reason, partial or targeted observation is more realistic and achievable. The observer can hope to reconstruct the boundary of an observer, or the neighbourhood of a chosen target, but not necessarily the entire topology.

An additional challenge is that some signals are indirect. Certain protocol messages, such as **GRAFT** and **PRUNE**, provide direct evidence of changes in the mesh. However, other observations provide only contextual and probabilistic information, such as messages, gossip traffic and peer-exchange. The observer must therefore

combine information sources and assess which are most likely to provide accurate information.

Finally, active participation in the network introduces an additional complication. An observer who joins topics and adds other observers to increase their visibility may disrupt the very network they are attempting to reconstruct. Thus, active observation can alter the topology that the observer is trying to model, thereby skewing the results. It is therefore necessary to measure this disruption and find a compromise so that active observation disrupts the observed network as little as possible. An attacker who maps out the network topology before launching an attack on a target has every reason not to disrupt the topology too much: if they do, they risk being detected.

Taken together, these challenges show that topology inference in GossipSub is not a simple graph recovery problem. It is a reconstruction problem under partial observability, temporal variation, protocol stochasticity, large scale, indirect evidence, and possible observer-induced perturbation.

2.3 From topology measurement to security implications

Topology is not a neutral and uniform element in a P2P network. The propagation of information depends on the paths, connectivity and position of the nodes. Certain nodes are more influential due to their key position within the network. If one controls the structure of a network, one can make assumptions about where the information will pass. One could identify the likely paths and provide a probabilistic propagation. By controlling the shape of the network, one has control over the information.

In the early days of P2P networks, the main concern regarding topology was performance. Gradually, knowledge of topology gave rise to another concern: that of security. Topology then becomes a surface of attack, allowing, for example, neighborhood manipulation [7].

In 2006, Singh et al. [7] published a paper introducing the concept of eclipse attacks on overlay networks. The aim is to fill a node’s neighbor table with Sybil nodes¹, thereby cutting the victim off from the rest of the healthy network. This is a purely structural attack, there is no need to break the cryptography or any other more complex layer.

Almost 10 years later, in 2015, Heilman et al. [8] applied the theory of these eclipse attacks to Bitcoin (a decentralized digital cryptocurrency based on blockchain technology [1]). Eclipse attacks have significant consequences for the network:

¹A Sybil node is a node controlled by a malicious participant.

double spending, transaction blocking, and propagation delays. In a network such as a blockchain, it is clear that topology is a critical element, as an attack on it can impact consensus and therefore security.

However, knowledge of the topology can make such attacks more severe. If one knows the network’s weak points, one can create a partition: the separation of the network into two sub-networks. With this knowledge, one can carry out strategic targeting. In this case, propagation is interrupted and a divergence in information may arise. The topology can therefore affect the consistency of the information.

This has been demonstrated on Bitcoin, which does not mean that Ethereum and IPFS are immune to it. These two networks, whose pub-sub layer operates using GossipSub, are also dynamic networks with an overlay and a local view. The challenges are the same, as are the potential vulnerabilities. In GossipSub, the possible attacks are well known [3], but few studies have examined the topology of GossipSub itself and its inference.

2.4 Security analysis of GossipSub

GossipSub was designed with security concerns in mind. However, it was not possible to account for everything. Certain threats are inherent in the very structure of a P2P network: the network is open, and potentially malicious peers may participate in it. Given that there is no central authority to verify everyone’s behavior, a node may behave maliciously or unpredictably in order to disrupt the network, as formalized by Kumar et al. [15] in their analysis of the GossipSub threat model.

There are multiple points of attack in GossipSub. Attacks can be launched via the topology by selecting suitable neighbors, either by knowing or by influencing the mesh structure. It is also possible to interfere with the dissemination of information by blocking or manipulating transmission. Another attack vector is to attempt to manipulate peer scoring to one’s advantage. In general, attacks revolve around the manipulation of mechanisms inherent to the protocol. Such attacks are, in fact, well known in GossipSub and are described in the paper by Vyzovitis et al. [3].

Kumar et al. [15] show, for example, that GossipSub’s scoring function can be configured in such a way that malicious peers manage to never relay messages while maintaining a positive score, thereby avoiding exclusion from the mesh. This observation is not unique to GossipSub: Pavloff et al. [16] observe a similar dynamic at the Ethereum consensus layer, where the inactivity leak² mechanism, designed to protect chain finality, can be strategically exploited by Byzantine validators. These

²The inactivity leak is an Ethereum consensus mechanism that gradually reduces the stake of validators who do not participate during non-finality periods, thus ensuring that the chain can eventually finalize even if a large portion of the validators disconnect.

two examples illustrate the same phenomenon: mechanisms designed to enhance a system’s robustness can, if misconfigured or misunderstood, backfire.

On paper, GossipSub is robust against attacks, but its robustness is probabilistic. It depends on the actual network. Topology plays a major role in this robustness, as does the proportion of honest peers. Many attacks rely on knowledge of the topology and control of the structure. With better knowledge, a more effective attack can be devised. Whilst attacks have been well studied, the topological reconstruction that precedes them remains largely unexplored in GossipSub.

2.5 Adversarial Model

An attacker can take various forms within the network. They may be a passive observer, simply collecting the information they see passing by. They may be an active participant, using their involvement to gather information and influence the network. But a malicious node is rarely alone, they operate as part of a coalition to combine information.

An attacker has all the same rights as a legitimate network participant, but also the same limitations. They can join the network, receive and send messages, and observe local traffic. However, they are limited to their local view and have no additional control.

Their goal is to reconstruct the topology sufficiently completely to identify key nodes and attack them accordingly. The objective of an inference may vary depending on the intended attack. It may be a local, targeted objective, such as reconstructing the neighborhood of one or more nodes, or a global one, aiming to cover as much of the network as possible.

The cost of an inference depends on the number of nodes involved in the inference. The cost increases with the number of nodes because machines are needed to host the nodes, as well as the necessary bandwidth.

Chapter 3

Background

This chapter will provide an overview of GossipSub, which we will be discussing at length later on from a technical perspective. The chapter begins with an introduction to the origins of GossipSub and continues with technical explanations.

3.1 Gossip-based dissemination protocols

Some P2P networks use FloodSub [17], a publish-subscribe protocol that employs naive broadcasting: it sends information to all neighbors, triggering a cascade effect. This results in a huge volume of messages and massive redundancy, and therefore a lack of scalability.

As networks have grown larger and larger, an alternative has had to be found: gossip protocols. These are inspired by epidemic models introduced notably by Demers et al. [18] with random dissemination. Each node contacts a subset, resulting in probabilistic propagation, as formalized in work on modern gossip protocols [19]. The dissemination of information is therefore more scalable. In a gossip model, complete dissemination is not 100% guaranteed, it is probabilistic. Redundancy is controlled. It is a trade-off between efficiency and reliability. Certainty is exchanged for high probability in exchange for a more scalable network.

It is in this context that GossipSub was created, networks are growing, and there is a need for control and robustness. GossipSub builds on gossip principles by introducing structured dissemination, it also mitigates the shortcomings of pure gossip with its hybrid approach.

3.2 Design and objectives

Based on previous research, we know that FloodSub is not scalable and is therefore unsuitable for large P2P networks [3]. However, pure Gossip is unreliable because it

is probabilistic [18], [19]. Neither of these approaches alone is sufficient to provide the necessary guarantees.

It is in this specific context that GossipSub, introduced by Vyzovitis et al. [3], steps in to act as a compromise between these two approaches, aiming to provide: scalability, reliability and robustness. It is therefore a multi-objective trade off. This compromise is achieved thanks to GossipSub’s hybrid design, which combines mesh-based transmission and gossip. It is probabilistic but structured. The protocol adheres to the basic principle of P2P networks: the view is local. And the algorithm’s decisions are made based on this local view. GossipSub also introduces a score to ensure network quality. With these features, GossipSub improves upon FloodSub and pure gossip by reducing redundancy and introducing structure.

GossipSub is used in production in critical systems such as Ethereum and IPFS, where the dissemination of information is central to the system’s operation.

3.3 Overview of GossipSub

GossipSub is a publish-subscribe protocol: some nodes publish information, and other nodes can subscribe to a topic to receive that information. All nodes play an active role in the network: the dissemination of information. In a GossipSub network, there may therefore be several different topics to publish to or subscribe to. In libp2p, the role of GossipSub is to ensure the efficient dissemination of messages between different nodes in a decentralized peer-to-peer network. This protocol is used, for example, in large blockchains such as Ethereum or IPFS.

It is important to distinguish between the physical network, which enables connections between different machines on a network (TCP connections), and the overlay, which refers to the logical connections between peers. GossipSub operates on this overlay [20]. The topology of GossipSub is a logical network superimposed on the physical network, it does not directly correspond to the actual network.

Before GossipSub, there were naive approaches to communication protocols: FloodSub. The principle is simply to broadcast the information to all neighbors as soon as a node possesses it. As the information propagates, it is sent to everyone. But this method raises several issues. Firstly, it consumes an enormous amount of bandwidth, most of which is wasted, as there is a great deal of redundant sharing. Redundant transmission increases with the size of the network. This leads to inefficient use of bandwidth [3].

It is in this context that GossipSub is introduced, it is a hybrid solution: direct broadcasting to a subset of neighbors of limited size (the mesh) for each node, and broadcasting via metadata to a few other neighbors (gossip) as shown in Figure 3.1. If a node receives metadata for information it does not have, it can send an `IWANT` message, an explicit request for targeted retrieval, which allows it to retrieve the

message in question [3]. Thanks to this, bandwidth usage is limited and efficiency is increased.

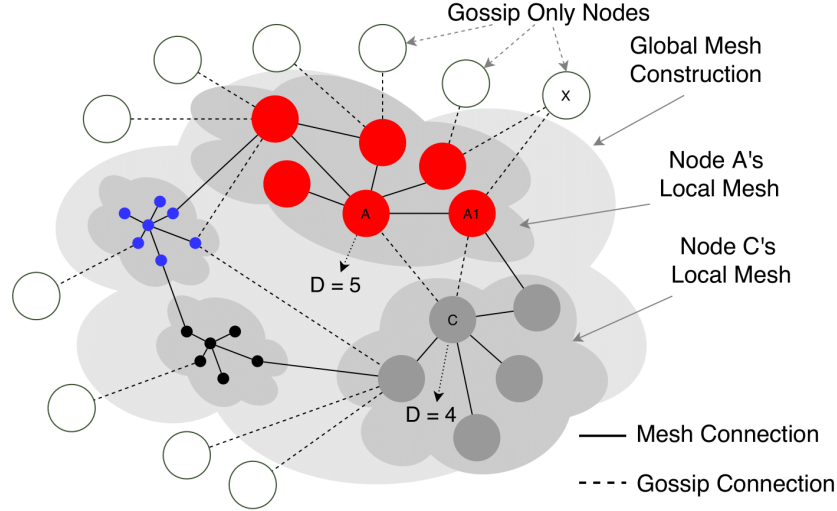


Figure 3.1: GossipSub Mesh¹

A major difference introduced by GossipSub is that delivery is not deterministic. Message delivery is not strictly guaranteed in a deterministic sense, it is designed to achieve a high probability of complete delivery [3]. GossipSub therefore offers a good balance between reliability and cost, providing controlled redundancy and thus better utilization of bandwidth. It is a more scalable solution, as each peer has a small neighborhood and therefore makes fewer transmissions. However, this balance is not guaranteed in all configurations. Recent analyses have shown that the effectiveness of GossipSub strongly depends on the tuning of its parameters, and that incorrect configurations may allow adversarial peers to degrade the protocol while remaining undetected [15].

To achieve these properties, GossipSub relies on several mechanisms.

3.4 Core mechanisms of GossipSub

To ensure secure and reliable communication, GossipSub employs several key mechanisms that determine how information is transmitted within a P2P network.

¹The illustration comes from a GossipSub presentation [20].

3.4.1 Topic-based Mesh Construction

GossipSub is topic-based, so each topic has its own broadcast structure: two topics that use the same physical machines do not necessarily have the same GossipSub overlay. The protocol does not broadcast information to everyone, instead, a node distributes the information to its local structure, known as a mesh. A mesh comprises a subset of neighbors chosen by a node to broadcast its messages. The aim of this structure is to limit the cost of broadcasting. A mesh is not a fixed structure: it evolves over time as it is maintained dynamically.

A node can subscribe to several topics, for each of these topics, it maintains a set of specific neighbors: its mesh. This means that two nodes may be neighbors for one topic but not communicate directly for another. A node's participation depends on its subscriptions. Distribution follows communities of interest.

A mesh is the set of peers to which a node directly sends complete messages for a given topic. These are 'active' neighbors in the main broadcast. The mesh is a structure that is local and specific to each node. This means that a node's mesh is not necessarily identical to that of its neighbors within the same topic. The overall structure of the network is the result of the aggregation of all the local meshes. As shown in Figure 3.1, there are mesh aggregations that form the overall structure.

The size of a mesh is not left to chance, it is controlled by parameters. The number of peers to which a node connects is called the degree. GossipSub has several parameters that control this degree. The most important is D [3], the target degree of the mesh. The aim is to keep it around D to ensure the neighborhood is neither too large nor too small. A mesh that is too small reduces redundancy, whilst a mesh that is too large increases the network's cost in terms of bandwidth. A compromise must be found between cost and robustness: using the right amount of bandwidth to ensure information is transmitted quickly and to everyone.

Then there is D_{low} [3], which is the minimum threshold for the number of neighbors. If the mesh falls below this threshold, it is considered too small and must recruit new neighbors. This prevents a node from remaining with insufficient connectivity.

After, there is D_{high} [3], which follows the same principle as D_{low} but for the upper limit. If the mesh exceeds this number, it is considered too large, so the protocol must reduce the neighborhood. This prevents uncontrolled growth in the number of transmissions. GossipSub does not seek a perfect global topology, it seeks a local balance around simply defined thresholds.

Next, there is the D_{out} [3], which checks the minimum number of outgoing connections from the node. In the network, connections are directed from the node that initiated the connection to the destination node. This direction makes it possible to monitor the incoming and outgoing connections for each node. It is important for a node to maintain outgoing connections because they allow it

to avoid becoming isolated among malicious nodes. By maintaining outgoing connections that it initiated itself, this ensures that each node has selected a subset of its neighbors, so it won't be isolated solely by these malicious nodes.

And there is D_score [3], which serves a more specific purpose. When the mesh exceeds D_high , meaning it is too large, D_score is used to determine which peers to keep by retaining the highest-scoring ones. This ensures that part of the mesh remains composed of peers deemed to be of good quality.

When a node does not have enough neighbors in its mesh, it must add some as shown in Figure 3.2. This operation is carried out via a control message called **GRAFT** [3]. **GRAFT** is used to signal that a node wishes to incorporate another node into its mesh for a given topic. Thanks to this mechanism, a node can restore a satisfactory degree in its mesh, it is triggered, amongst other things, when the mesh falls below D_low .

Conversely, when a node has too many neighbors in its mesh, it must remove some as shown in Figure 3.2. This operation is carried out via another control message called **PRUNE** [3]. **PRUNE** is used to notify a peer that it will no longer be part of the mesh for a given topic. This mechanism allows the mesh to return to a controlled size when it exceeds D_high .

GRAFT and **PRUNE** are network control messages, with **GRAFT** being a request to join and **PRUNE** a notification of removal.

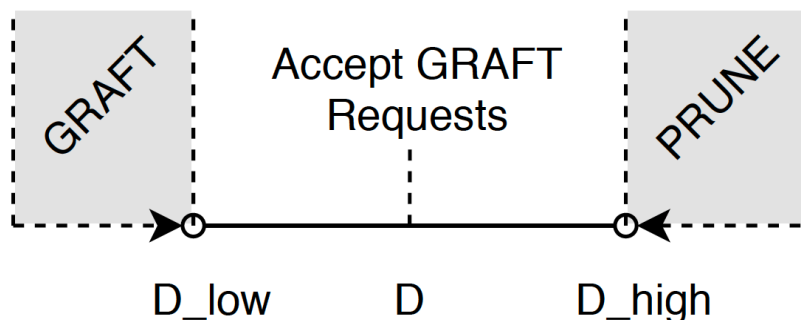


Figure 3.2: Ideal degree for a mesh²

It is useful to understand that connections have a direction: when one node sends a **GRAFT** to another node, the connection points towards the node that received the control message. This direction is important for maintaining a mesh.

It is important to note that the mesh is maintained locally, not globally. Each node makes decisions based on its own local view. There is no central authority or global coordinator. Consequently, the topology is not calculated globally, it results from the aggregation of local decisions: it is a decentralized system.

²The illustration comes from a GossipSub presentation [20].

The mesh is a dynamic structure over time. Its composition evolves with changes in the network. neighbors may be added and removed over time, this is a normal dynamic in GossipSub, linked to periodic maintenance mechanisms: the network’s heartbeat.

3.4.2 Message propagation mechanisms

GossipSub does not rely solely on the mesh principle, it combines direct broadcasting within the mesh with indirect broadcasting known as gossip. These are two complementary mechanisms designed to ensure efficient propagation whilst minimizing network overhead.

The first form of propagation is the direct propagation of a message. When a node receives a message, it transmits it to its neighbors within its mesh. Each neighbor does the same with its own mesh, so the message propagates from one node to the next. The mesh serves as the broadcasting infrastructure. There is a natural redundancy associated with the multiple paths of this broadcast. A message sent via direct propagation is sent in its entirety.

However, a mesh does not cover the entire network directly, if we rely solely on this direct mesh transmission mechanism, propagation delays may occur. Transmission is highly dependent on the network’s structure: if two parts of the network are poorly connected, the message may take some time to arrive.

It is within this context that GossipSub’s second dissemination mechanism comes into play: indirect dissemination. This is based on metadata: the aim is to inform nodes other than those in the mesh without sending the full message. This is what is known as ‘gossip’.

To do this, when a node receives a message, in addition to sending the full message to its neighbors within the mesh, it selects a subset of nodes outside the mesh and sends them a control message called **IHAVE** [3]. This message contains the identifier of the message in question. Nodes that receive an **IHAVE** message therefore know that a message exists.

When a node sees an **IHAVE** message, it checks whether it has already received the message in question or not. If not, it responds with another control message: **IWANT** [3]. The **IWANT** message allows a specific message to be requested from the node that sent the **IHAVE** message. This process is illustrated in Figure 3.3. This mechanism therefore enables efficient yet controlled propagation across the network: a node only requests what it needs and what it has not yet received. It is through this mechanism that bandwidth reduction occurs, as unnecessary duplication is avoided.

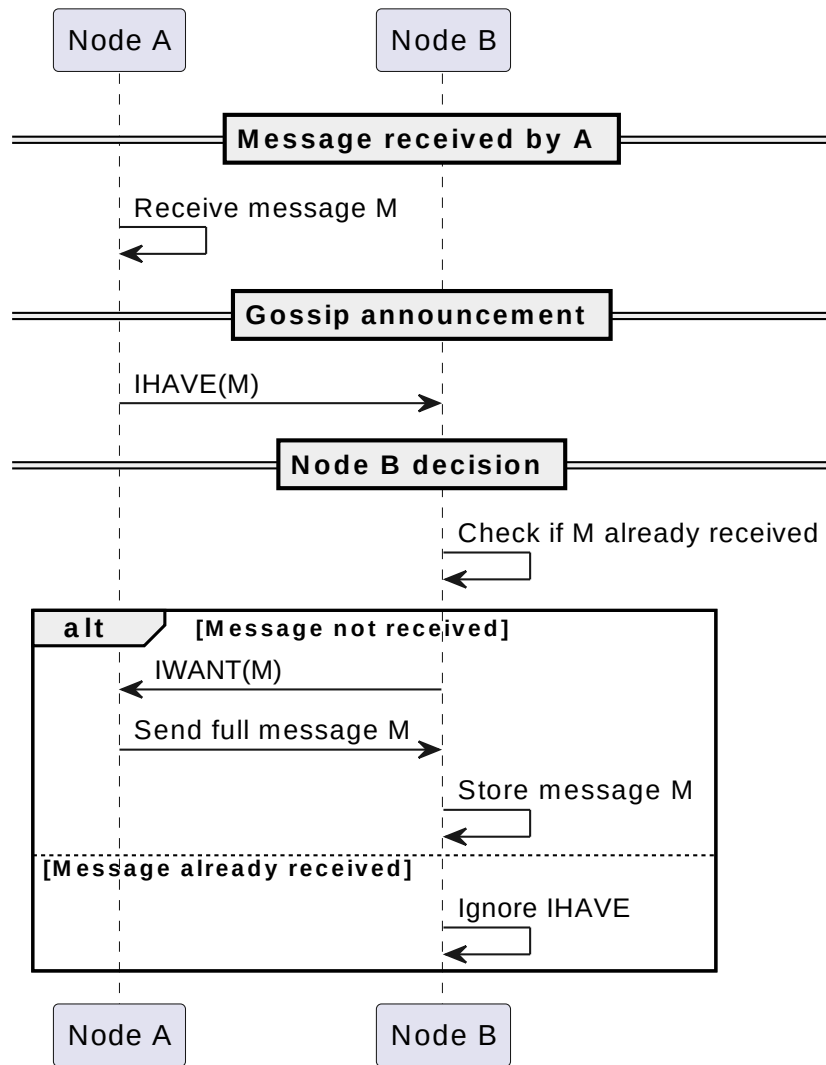


Figure 3.3: GossipSub IHave / IWANT Interaction

The two mechanisms work hand in hand to broadcast messages within a network. Direct broadcasting within the mesh is the primary mechanism, whilst gossip serves as a fallback mechanism. When the mesh is functioning properly, gossip is rarely used, but in the event of a problem, it is there to rectify the situation. In terms of the overall topology, this improves robustness and loss tolerance, thereby ensuring better coverage.

3.4.3 Peer scoring system

In a decentralized network open to everyone, it is to be expected that not all peers will behave in the same way. Some nodes may relay messages correctly, whilst others may be unreliable or behave incorrectly. The protocol must therefore have a means of evaluating the behavior of each node: scoring. The score influences the way a node interacts with its neighbors. Ultimately, the aim is to improve the quality of the broadcast.

The score is not a global concept within the network, each node evaluates its neighbors from its local perspective and maintains a score for the neighbors it knows. The score is calculated locally. This means that two different nodes may assign different scores to the same peer. The evaluation depends on local experience and interactions observed by the evaluator. This logical choice is consistent with the operation of a decentralized network, where there is no single trusted authority.

The score is used to rank peers within a mesh and identify the most useful and reliable ones. It influences the selection of neighbors within the mesh. If a peer needs to be removed from the mesh, the one with the lowest score is chosen. Conversely, peers with the highest scores are given priority in certain protocol decisions. The quality of the neighborhood therefore depends largely on the score, the composition of a mesh is not arbitrary. The score thus plays a part in the composition of the mesh.

The score is the result of several components corresponding to different aspects of a peer's behavior within the mesh. Some reward good behavior, whilst others penalize undesirable behavior. The final score is the result of weighting all these components using a formula of this type, which represents a weighted combination of the various components of the score [3], [15].

$$\begin{aligned}\text{Score}(\text{peer}) = & w_1 \cdot P_1 \\ & + w_2 \cdot P_2 \\ & + w_3 \cdot P_3 \\ & + w_{3b} \cdot P_{3b} \\ & + w_4 \cdot P_4 \\ & + w_5 \cdot P_5 \\ & + w_6 \cdot P_6 \\ & + w_7 \cdot P_7\end{aligned}\tag{3.1}$$

In the Equation 3.1, each component P_i corresponds to an aspect of the score, which we will look at below, whilst w_i is the weight associated with each component.

The first component is the time in the mesh (P_1). The more stable a peer is and the longer it remains in the mesh, the greater the reward. However, there is a cap:

after a certain amount of time, the reward no longer increases. The other reward is first message deliveries (P_2). This rewards peers who relay messages quickly. It encourages rapid propagation.

Next, there are two participation components. The first is mesh message deliveries (P_3). This measures the number of messages sent by a peer within the mesh, to check whether they are sending enough. If the peer does not relay enough within a given time frame, they receive a penalty. And there is the Mesh failure penalty (P_3b): if a peer leaves the mesh whilst not participating sufficiently, they are penalized. This discourages opportunistic behavior.

After that, there are the penalties. The first penalty is the invalid messages penalty (P_4). The aim is simple: to heavily penalize malformed or incorrect messages in order to maintain network quality. Another penalty is the behavior penalty (P_7), which penalizes suspicious behavior (spam, flooding, etc.).

Finally, there are two other components. The application-specific score (P_5), which allows the application using GossipSub to add its own weighting to the score for greater flexibility. And the IP colocation factor (P_6), which allows multiple peers on the same IP address to be penalized.

Weighting is very important, not all components carry the same weight, so a single bad action can outweigh several good ones. It is important to incorporate the concept of time into the score. Scores evolve over time. Some components decay automatically over time: a good score is not maintained by inactivity. A peer must keep performing good actions, otherwise its score declines.

Not all components of the score are linear, and significant penalties are applied if performance falls below a given threshold. This is the case, in Equation 3.2, with the component relating to the number of messages relayed within the mesh.

$$P_3 = \begin{cases} (\text{threshold} - \text{deliveries})^2 & \text{if deliveries are below the threshold} \\ 0 & \text{otherwise} \end{cases} \quad (3.2)$$

Since P_3 is positive, its associated weight w_3 is negative in GossipSub. This will therefore result in a deficit in the node's score. This formula ensures that a progressively higher penalty is applied when the number of messages relayed falls below a threshold, but no penalty is applied if the threshold is met.

GossipSub uses score thresholds. Below a certain level, certain interactions are prohibited or restricted. Above this level, a peer is eligible to continue their activities normally within the network. These thresholds are what enable continuous evaluation to be translated into decisions.

The score evolves over time, it is updated as interactions take place. It reflects behavior over time. If a node has a poor score but behaves well for a period, its score will rise. A ranking, whether good or bad, is not definitive.

3.4.4 Heartbeat and mesh maintenance

GossipSub is not based on a static structure, the network is constantly evolving. To ensure this evolution, the protocol uses a periodic mechanism known as the heartbeat. This mechanism is executed locally by all nodes at regular intervals, and it enables the mesh to be maintained and adjusted.

The heartbeat is executed at regular intervals, with each node having its own cycle. It is important to understand that there is no global synchronization, once again, it is the sum of local decisions that determines the overall topology. The most common frequency is 1 second [3]. It is a multi-step loop: observe, decide, act.

The first element checked by the heartbeat is the size of the mesh using the three parameters D , D_{low} and D_{high} [3]. If the mesh is too small (below D_{low}), the heartbeat will use **GRAFT** to add peers to the mesh. Conversely, if the mesh is too large (above D_{high}), the heartbeat will use **PRUNE** to remove the peers with the lowest scores from the mesh. The heartbeat will ensure that the mesh centers around D , the ideal degree.

The heartbeat will also ensure that there is a minimum number of outgoing connections via D_{out} [3] so that the node does not end up surrounded by nodes that control the information passing through it.

The heartbeat not only monitors the number of peers, but also the quality of the mesh. It may decide to gradually replace the peers with the lowest scores in its mesh, in order to improve its quality over time. The selection of new peers is based on their score and availability. This mechanism is called Opportunistic **GRAFT**. It improves the mesh qualitatively, not quantitatively. A peer may know of other peers that can improve its mesh through discovery systems outside of GossipSub, and it may already have a good impression of those nodes, for example, because it has previously exchanged gossip with them.

As mentioned previously, the cornerstone of selecting new peers or peers to remove is scoring. The heartbeat's decisions are based primarily on this, whether for selections or removals.

The heartbeat is not merely a mechanism for maintaining the quality of the mesh, it also serves other functions: sending **IHAVE** messages, internal cleaning, updating local states, etc.

Thanks to the combination of scoring and the heartbeat, from a global perspective, the sum of local decisions improves the quality of the network and ensures efficient dissemination.

3.5 Additional protocol mechanisms

To enable the network to function, GossipSub implements secondary mechanisms that optimize the network and improve its performance.

3.5.1 Peer exchange (PX)

One such mechanism is Peer Exchange (PX) [3], which allows information about other peers on the network to be shared in specific situations. When a node sends a **PRUNE** to another node, effectively removing it from its mesh, it can provide that node with a list of other peers. This helps prevent isolation and maintain connectivity. This list comprises alternative peers that are potentially relevant and connectable. It is a filtered list of limited size.

PX is not a core peer discovery mechanism that occurs all the time. In the GossipSub v1.1 specification, it is an extension of the **PRUNE** message [21], designed to help the peer being **PRUNED** to rebuild its neighborhood. This avoids the need to rely on another peer discovery service for mesh formation.

Note that when a peer is **PRUNED**, it enters a backoff period. Both peers involved in the **PRUNE** record a backoff period during which they cannot **GRAFT** to one another [20].

Not all **PRUNE** messages result in a PX, the two most common situations are when a node attempts to **GRAFT** someone else's mesh but that mesh is already full. In this case, the node that owns the mesh may reject the **GRAFT** with a **PRUNE** and attach a PX list to it. A second case is when a node's mesh becomes too large, so it wants to remove a pair from its mesh. It will send the pair a **PRUNE** message with a PX list so that it can rebuild its mesh. The process for the second case is illustrated in Figure 3.4, where a node, having received a **PRUNE** message from another node, uses the attached PX list to **GRAFT** a new node.

A PX list is limited to 16 peers per list. The list is constructed by the node sending it at the time of transmission, it is not pre-constructed. This means that two PX lists sent by the same node will not necessarily contain the same peers.

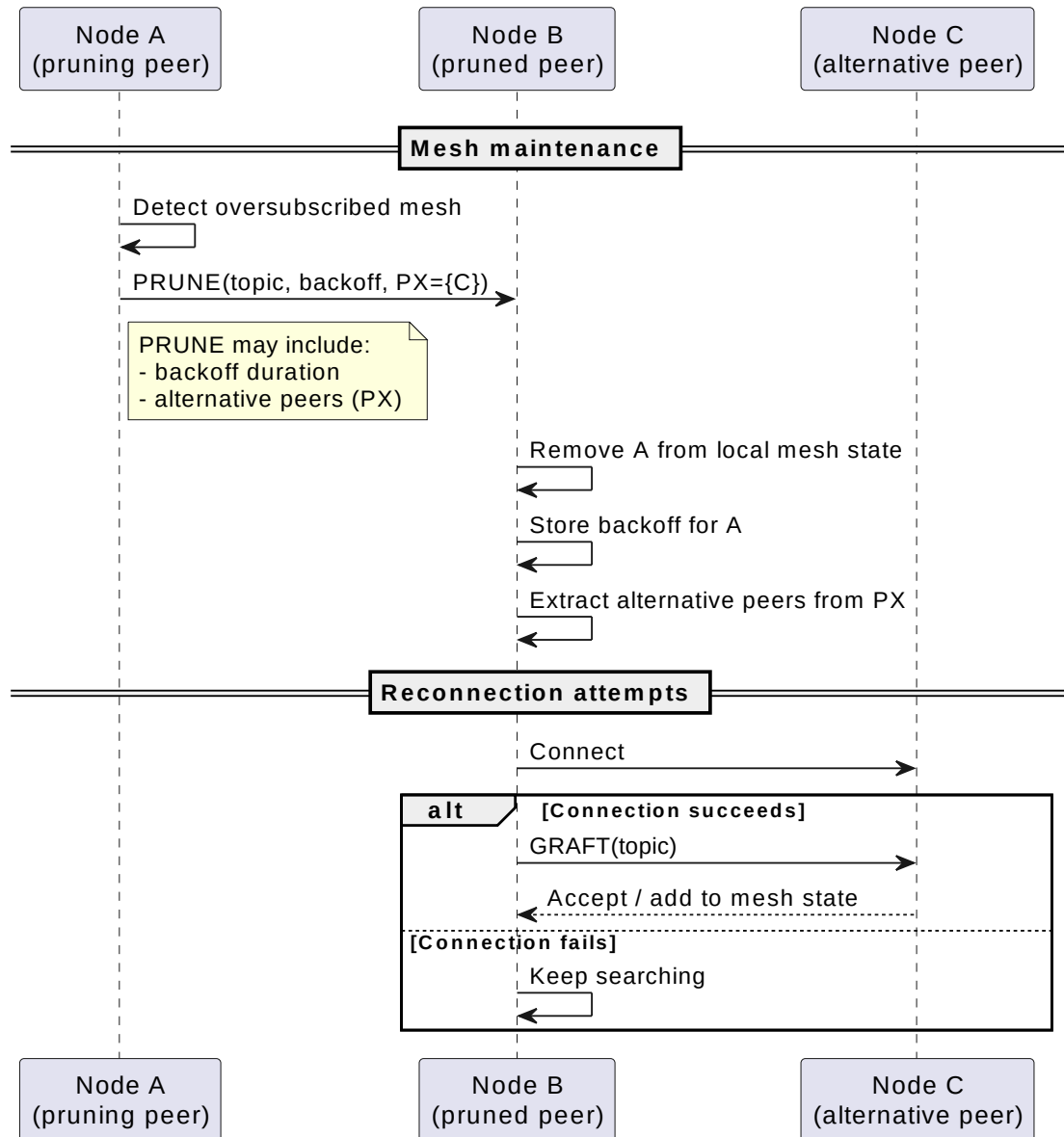


Figure 3.4: GossipSub PRUNE + PX Interaction

The PX mechanism is an optional mechanism that is not, and should not in a normal configuration be, enabled on all nodes in a network. It is intended to be enabled on bootstrap nodes or trusted/well-connected nodes.

Often, applications that use GossipSub rely on discovery services external to GossipSub, rendering systems like PX unnecessary or unused.

To build the contents of a PX list, a node draws on nodes from various sources. These can be nodes from its own mesh, nodes to which it is connected but which

are not in its mesh, for example, nodes with which it simply exchanges gossip. They can also be nodes discovered through an external discovery system. The key point is simply that they must be connectable nodes.

3.5.2 Peer discovery

This subsection will address an issue that does not depend solely on GossipSub, but which is important to understand for what follows. GossipSub is not designed to provide a complete peer discovery system [3]. Instead, it assumes that nodes already have a minimal set of libp2p connections. The only way to obtain peers directly through GossipSub is via Peer Exchange, which was discussed earlier. Otherwise, external discovery mechanisms must be used. With this option, a node uses an external protocol to find libp2p peers, and then GossipSub can select from among these nodes to form its mesh. A common example is Kademlia [22], where the node queries a DHT (Distributed Hash Table) to obtain peers associated with a topic. We can also mention discv5 [23], which is used in the modern Ethereum network.

Chapter 4

Simulator

To study the topology, tests must be carried out on the GossipSub protocol. However, several methodological problems arise. On the one hand, on a real network, the topology is difficult to study because it is not observable: a participant has only a local and partial view of the network. Furthermore, it is not possible to conduct repeated experiments under identical conditions, which complicates comparisons. Nor is it possible to isolate the various components of the protocol, such as scoring, pruning or peer exchange, in order to assess their role in the formation of the topology. We have no control over the network, we can only participate in it and draw conclusions from what we observe. For ethical reasons, we cannot actively disrupt the network too much for our tests either.

To address these issues, this thesis makes use of a dedicated simulator¹. The simulator employed is an enhanced version of an existing GossipSub simulator [11]. This simulator addresses these various issues. It provides a controlled environment in which the protocol parameters and initial conditions can be explicitly set. Thanks to this, runs can be reproduced exactly using identical seeds. Certain GossipSub mechanisms can also be enabled or disabled to better understand the causality of certain results. And, most importantly, we have access to the true topology, so we can compare an inferred network with the actual network. Another advantage of the simulator is that it requires fewer resources.

It is important to emphasize that the simulator is used to study GossipSub's topology inference and the causes of this topology. The simulator is not intended to faithfully reproduce the full complexity of a real libp2p network. The simulator is topology-oriented, as required by this thesis.

¹<https://github.com/V1rg1lee/kademlia-simulator/tree/gossip-complete>

4.1 Design objectives

When improving this simulator, we had three objectives in mind:

- Topological fidelity
- Lightweight performance
- Realism

The simulator does not aim to maximize all three factors equally, but seeks to be lightweight enough to run campaigns within a reasonable time-frame, whilst remaining realistic enough to produce credible inferences that are faithful to the topological dynamics of GossipSub.

Of the three, topological fidelity is the priority, as the ultimate goal is topology inference. Consequently, any mechanism that influences the topology is therefore a potentially interesting mechanism to replicate in the simulator. There are several key mechanisms:

- mesh formation and maintenance,
- topic separation,
- heartbeat-driven mesh evolution,
- GRAFT and PRUNE decisions,
- scoring and its influence on peer retention or rejection,
- opportunistic GRAFT,
- local peer states such as known, connectable, connected, and mesh membership.

By contrast, several parts of a full libp2p deployment are deliberately abstracted away because they are not central to the topology inference problem considered in this thesis. These include, for example:

- multiaddrs and address management,
- fine-grained latency modeling,
- NAT traversal,
- transport-layer details and full connection establishment semantics.

This is not a full libp2p simulator, but a topology-oriented simulator designed to model GossipSub.

4.2 Simulation environment and performance

PeerSim is a Java-based event simulator that runs as a single-threaded process, in which each network node is an independent Java object. The following metrics were measured on an x86-64 processor running Ubuntu 24.04 with OpenJDK 17.0.18 for a PX flood inference run described later: the most memory and process intensive inference.

The **run.sh** command allocates up to 300 GB of JVM heap (-Xmx300000m), but this is a theoretical limit, not actual consumption. In practice, the JVM uses the memory it deems necessary.

In all our future experiments, we run each test on three different seeds. Performance tests are therefore also conducted on these three seeds to ensure consistency. Here are the various figures calculated based on the seeds, it is important to show the variability among the three seeds on the simulator network because these will be the three seeds used throughout the entire campaign. The test results are shown in Table 4.1.

Table 4.1: Simulation performances

Seed	Real time	CPU time	Ratio	RSS peak ²
24680	50s	46s	0.92	6.0 GB
13579	54s	48s	0.89	6.7 GB
98765	50s	47s	0.94	4.7 GB
Mean	51s	47s	0.92	5.8 GB

Given that we have 1,000 nodes, approximately 5.8 MB is required per node. Each simulated node maintains in memory its GossipSub mesh, its table of candidate connections, as well as the associated scoring metadata, event buffers, inference registers, and message scoring windows. A few MB is therefore reasonable for such a rich protocol.

We observe variability in the RSS among the seeds: 4.7-6.7 GB. This is not noise, different seeds can produce different topologies. If a topology has more edges, more Java objects are allocated, thus requiring more memory.

The CPU time-to-real-time ratio of 0.92 indicates that the JVM occupies 92% of the core’s real time. It almost never waits because there is very little I/O and no sleep. The ratio is below one because it is single-threaded. But it is close to one because we are running it at full capacity without waiting.

Memory usage is an implementation-specific property and remains consistent on any other x86-64 machine using the same JDK. However, execution time may

²Resident Set Size refers to the maximum amount of RAM (physical memory) used by a computer process during its entire execution.

depend on and vary depending on the CPU.

The measurements were obtained using `/usr/bin/time -v` applied directly to the JVM process (`run.sh`), which reports the peak RSS of the process tree as tracked by the Linux kernel. The exact command is documented in the repository README³.

4.3 Architecture

The simulator’s architecture has been designed as a multi-layered model to meet all the requirements of the experiment, ranging from scenario configuration to the generation of results for analysis. It is therefore not merely a protocol execution engine, but an integrated tool that allows users to simulate the dynamics of GossipSub, extract a topological ground truth, and evaluate various inference observations.

The simulator’s operation thus consists of a sequence of several layers.

The first layer is the experimental configuration. It defines the various protocol parameters, the scenarios under study, the seeds used for reproducibility, the profiles of the various nodes, including the observers, and so on. This layer precisely sets the initial conditions for a simulator run and controls which mechanisms are or are not activated during the experiment.

The second layer is the protocol’s simulation kernel. Each simulated node has a local state that represents its local view of the network and its GossipSub behavior. This state includes, in particular, its peers by topic, information about the various peers, the parameters that influence its local decisions, and other details. The simulator is based on a distributed model: there is no global view of the protocol used to direct the nodes, rather, it is a collection of local actions that interact with one another to form the global graph. This is a key property because GossipSub is decentralized in nature, and it lies at the heart of the inference problem studied in this thesis.

The system’s evolution is ensured by periodic maintenance logic, organised by the global heartbeat. At regular intervals, each node performs maintenance operations on its topics: increasing a mesh that is too small, applying scoring rules, opportunistic **GRAFT**, propagating gossip, and verifying the number of outgoing neighbors. The heartbeats provide a temporal unit for the simulation. The heartbeat is not only useful for network maintenance, it also structures the observation of the network by providing consistent reference points for extracting the topology. This synchronization of heartbeats is an abstraction created by the simulator, in a real GossipSub network, the nodes are asynchronous with one another.

³https://github.com/V1rg1lee/kademlia-simulator/blob/gossip-complete/simulator/topology_inference/README.md

The next layer is used for data logging. It records two types of information. Firstly, it logs topological events, such as the addition and removal of mesh neighbors, joins and leaves. Secondly, it produces complete snapshots of the mesh's state at specific points in time. A distinction must be made between these two types of information. The events enable the analysis of GossipSub's health and operation, and help to understand the protocol's dynamics. The snapshots serve as ground truth for evaluating inferences. This recorded data is not accessible to observers attempting to infer the topology, it is a view reserved for the evaluator to compare the inferred topology with the actual simulated topology.

There is also a layer for collecting and exporting results. When a run ends, the traces generated during the simulation are serialized into usable files. These contain all the information needed for subsequent processing: exchanged messages, topology events, mesh snapshots, aggregated run metrics, and inference-specific outputs. This is a very important layer of the architecture as it enables the transition from simulation to data analysis. The simulator is not designed to be observed whilst running, but to be analyzed after execution based on the traces it leaves behind.

And finally, there is a final layer: the experimental and analytical layer. This layer is implemented through scripts for campaign launch, data aggregation and data evaluation. Thanks to this layer, part of the process is automated, and experiments can easily be repeated across multiple scenarios, multiple seeds, multiple observation budgets and multiple coalition sizes. This layer reconstructs the ground truth from the snapshots, reads the inferred topologies from the observers, calculates the metrics, and aggregates the results for the entire campaign. The architecture therefore comprises not only the simulator but also the measurement and comparison of data, which enables a run to be transformed into an interpretable result.

This architecture has several advantages. Thanks to the separation of layers, there is a strict distinction between the local view of the observer attempting to infer and the global ground truth used for evaluation. This separation is important for the credibility of topology inferences: observers can only use the signals they observe locally, whereas the evaluator has access to the snapshots. This architecture also promotes reproducibility, given a configuration and a seed, one can regenerate an identical run with its final traces and metrics.

The architecture has also been designed to allow for different types of monitoring (passive or active) without requiring any changes to the simulator itself. All monitoring relies on the same technologies and tools. What varies from one inference to another is primarily the nature of the monitoring and the decisions that are made.

4.4 Modeling of GossipSub

The aim of this section is not to explain in full how GossipSub works, as this has already been covered in the background chapter. Its purpose is to explain how the protocol mechanisms that influence the topology are implemented in the simulator. In the context of this research, the aim is not to reconstruct the entire software stack of a libp2p deployment, but to model the topology faithfully enough to draw conclusions from it.

The first point to note about this modeling is that the GossipSub topology is not a global structure shared by all participants. Being decentralized, it is a collection of local relationships, maintained by each node for each topic. The simulator incorporates this fundamental property. The topology is therefore not represented in the simulator as a single graph, but as a set of local mesh relationships, directed and assigned to a topic. This is very important because this property lies at the very heart of our research problem. When a node maintains a pair in its mesh for a topic, this is a local relationship, but one that is not necessarily symmetric. The simulator therefore considers that a topological edge signifies that a node locally regards another node as a member of its mesh for a given topic.

This local, topic-based representation forms the basis of the simulated topology's dynamics. Thanks to this distinction by topic, if a node is in a mesh for one topic, it will not necessarily be in the same mesh for another topic. This allows us to study not only the topology but also the position of nodes within different topics and their importance relative to those topics.

The topology of GossipSub is one that evolves over time. This evolution is driven by a mechanism of periodic maintenance provided by the heartbeat. In the simulator, the heartbeat is the temporal unit around which the mechanisms are organized. It is at the time of the heartbeat that local structures are re-evaluated, repaired or modified. The heartbeat here has a dual role: that of maintaining the meshes, and that of providing a consistent temporal reference for experimental observation.

From a topological perspective, the heartbeat is the moment when the node evaluates its mesh and identifies its flaws: too small, too large, or of insufficient quality. It can therefore make adjustments based on local criteria. The simulator replicates this logic by making the heartbeat the moment when decisions regarding mesh maintenance are taken. These decisions include: repairing underpopulated meshes, reducing overpopulated meshes, applying selection rules between different competitors to join a mesh via scoring, and rules for adapting the local composition of the mesh to maintain a minimum quality. The heartbeat plays a central role: the aim is to infer the topology, and to do so we need a topology that behaves like the real implementation. It is important to note that in the real implementation of GossipSub, the heartbeat is initiated locally by each node. Here, to simplify

the network organization, a global heartbeat triggers each local heartbeat. This simplification preserves the local maintenance logic of GossipSub, but abstracts away the asynchronous scheduling of heartbeats that would occur in a real deployment.

When a simulation starts, there is a warmup phase that lasts for 10 heartbeats. At the very beginning, all meshes are empty. The first few heartbeats are therefore used to send **GRAFTs** so that each node can populate its mesh. This transition phase is not representative of a stable operational network, so no inference is performed during this phase. During the warmup, nodes are assisted by a mechanism to quickly find other nodes and allow the network to form. This mechanism is no longer available after the warmup. Ten heartbeats is longer than it actually takes for the network to stabilize, the network is stable after fewer than five heartbeats.

After the warmup period, nodes still have access to a mechanism for discovering new nodes. Every 10 heartbeats, each node can request to discover a new peer in the network. This system is intended to replace the frequent use of external discovery systems by applications that use GossipSub. It allows each node to have access to enough nodes to keep their mesh at an acceptable quality, without each node having access to a global list of nodes, which would not reflect the reality of GossipSub. This mechanism is called ambient discovery⁴.

Next, various more specific mechanisms were included in the simulator as they are also an integral part of the topological dynamics. Firstly, the **GRAFT** and **PRUNE** protocol messages. These are modeled as significant topological events, as they enable edges in the graph to be created or removed.

Scoring has also been included and can be enabled or disabled depending on the simulation’s requirements. Its role is not to exactly replicate GossipSub’s scoring, as this can vary depending on how GossipSub is implemented, since GossipSub leaves a parameter exploitable at the application level for the score. Its aim is rather to represent that certain peers are locally preferred, retained or rejected depending on the quality of each one’s behavior. A mesh therefore does not evolve solely according to a size constraint, but also takes into account the quality of the nodes within the mesh. The score creates a local hierarchy that allows nodes to be preferred when choosing between several. Scoring is an important mechanism as it has a significant impact on the stability of the mesh. The simulator therefore replicates the scoring, allowing for the comparison of scenarios where scoring is absent or not.

Following this line of reasoning, opportunistic grafting is also modeled in the simulator as a mechanism that changes the local topology when a node believes it can improve the quality of its mesh. This is an interesting mechanism as it

⁴This mechanism is an abstraction of the simulator with respect to GossipSub, designed to replace an external peer discovery system such as Discv5 or Kademlia.

introduces additional churn⁵ and results in higher-quality meshes.

However, the scoring system as currently implemented has limitations. Degraded nodes are marked as degraded and have a low score. Their score is not linked to their poor performance. This limitation therefore underestimates actual churn compared to a real network, because once degraded peers are removed from the mesh via scoring and heartbeats, the remaining peers will eventually perform equally well. In PeerSim, every communication between nodes has the same latency, making it impossible to score based on that. Due to this limitation in scoring, opportunistic **GRAFT** is also limited.

Some mechanisms do not represent a topological change but are nonetheless essential to include for the simulator and experiments to function properly. Among these mechanisms are propagation messages such as **IHAVE**, **IWANT** and **MESSAGE**. They act as activity signals and enable the network to function normally using the hybrid logic of gossip and messages.

The same applies to peer exchange, which can be enabled or disabled. It is also of interest because it has an indirect influence on the evolution of the topology and provides information about it.

Beyond these mechanisms, the simulator makes an important distinction between several local states of a peer. The simulator does not merely recognize two states for the relationship between two nodes: ‘in the mesh’ and ‘out of the mesh’. It recognizes more precise states, thanks to which a peer can be known locally, considered connectable, connected, or admitted into the mesh. This is an interesting distinction because in GossipSub, a node does not only have information about the nodes in its mesh but also about others via these intermediate states.

In contrast to the above elements, certain mechanisms have been deliberately omitted from the simulator’s implementation. The aim was not a real libp2p deployment. For example, detailed management of multi-addresses, network transport, NAT, fine-grained latency, the possible diversity of client implementations, etc. These elements may influence the network, but represent too much complexity and are not relevant in our case. A finely reproduced local topology is required, but not an exact reproduction of the protocol in all its aspects.

4.5 Ground-truth

To perform an evaluation, the evaluator needs two things: a truth and a subject to evaluate. In our case, the truth is a series of network snapshots. The subject to evaluate, meanwhile, is the topology inferred from the observations.

⁵In GossipSub, churn refers to the ongoing process by which peers join, leave, or change connections within the overlay network over time.

The topology snapshots used for evaluation are obtained by directly inspecting the internal state of the GossipSub simulator. They are a simulator tool that serves as ground truth, they are not reconstructed from messages observed by observers.

The mechanism is triggered within the global GossipSub heartbeat. At each heartbeat, a function traverses all nodes in the network. For each active node, we wait for the completion of its own heartbeat. Once all nodes have executed their heartbeat n , we read their current internal state. In other words, the snapshot associated with heartbeat n represents the state of the mesh after the heartbeat has been fully executed.

For each active node, the simulator reads all its edges for each topic. Each relationship of each node is recorded in the form of a directed edge and sorted by topic.

$$(\text{topic}, \text{node_id}, \text{peer_id}).$$

It is indeed the directed edges that are recorded. Thus, if node A contains node B in its mesh for a given topic, the snapshot will contain the entry $A \rightarrow B$ for that topic. Similarly, if node B contains node A in its mesh for a given topic, the snapshot will also contain the entry $B \rightarrow A$ for that topic.

Snapshots are accumulated in memory during the simulation. At the end of the simulation, the snapshots are serialized into a file. This file contains columns such as heartbeat index, topic, node id, peer id, mesh size.

The frequency of snapshot taking depends on the experimental mode. Depending on the inference being tested, snapshots can be triggered at every network heartbeat, or taken at defined time intervals. In our case, a snapshot is taken at every heartbeat.

It is important to note that the copies created are deep copies, ensuring that the data is not affected by subsequent changes. It is therefore a true snapshot of the data at that point in time.

At the very end of a simulation, a final snapshot is forced to capture the final state of the network. This feature ensures that the final metrics extracted are indeed those from the end of the simulation.

This mechanism provides the evaluator with reliable ground-truth data for the simulated network, but it is important to understand that it does not reflect the observer’s ability to infer the network. The observer can only see the messages and interactions between its nodes, it does not have access to the overall internal state of the network. Snapshots are only used offline to measure the quality of the inferences.

4.6 Fidelity assessment

A validation of the simulator was carried out to compare the GossipSub simulator with an actual run of libp2p/GossipSub⁶ on Grid5000 [24]. The repository used for the real deployment is a fork that come from an implementation of PANDAS [14]. The aim is not to achieve perfect equivalence edge by edge, as this would not be relevant. Rather, the aim is to verify whether the simulator correctly reproduces the observable structural properties of the GossipSub mesh. To do this, various topological metrics are used.

4.6.1 Metrics

To assess the similarity between the simulator and the actual GossipSub implementation, various metrics are used. These are primarily metrics that evaluate the structure of a network.

Degree, Density. The degree is the number of mesh neighbors of a node. Here we are interested in the out-degree, i.e. the number of peers that the node maintains within its mesh. This measure provides further insight into the effect of the parameters D , D_{low} and D_{high} . Density measures the proportion of edges present relative to the maximum possible number of edges in the graph. This measure provides insight into the connectivity of the mesh, if a graph is dense, the nodes are strongly connected to one another.

Connected components, diameter, average path length, clustering, giant. For the following metrics, it is important to understand the concept of connected components. Two nodes are in the same connected component if a path exists directly between them or via other nodes. This measure helps to determine how many subgraphs exist within the overall network. The diameter corresponds to the greatest distance between two nodes in the same connected component. It shows how large the network is, or whether it is highly subdivided. The average path length measures the average distance between connected nodes. Clustering measures the tendency for a node's neighbors to be connected to one another as well. High clustering indicates a strong tendency towards a redundant network of meshes. The giant component gives the number of nodes in the largest connected component.

Churn, temporal stability. With regard to temporal dynamics, we use churn and temporal stability. Churn measures the proportion of mesh edges that change

⁶<https://github.com/V1rg1lee/GossipSub-Minimal>

from one snapshot to the next. It is an indicator of the extent to which the graph reorganizes itself over time. A high churn value indicates that the graph changes significantly over time, meaning the topology is more unstable. Finally, temporal stability is calculated using Jaccard similarity (described below in section 5.1.5) between two successive snapshots. This measure compares all edges at two points in time. If it is close to 1, the topology has changed little, conversely, if it is close to 0, the topology has changed significantly.

These metrics are suitable for our study as they allow us to assess whether the topology is plausible even if the details of local decisions vary from one run to another.

4.6.2 Context

The actual experiments were run on three different scales: 20, 50 and 100 nodes. These scales were chosen to enable the metrics to be tested across different sizes, taking into account the reasonable maximum number of nodes that could be tested on Grid5000 for this work. In each case, 20% of the nodes are degraded, i.e. 4 out of 20, 10 out of 50 and 20 out of 100. The real GossipSub uses a single topic per experiment, with peer scoring enabled. The GossipSub parameters on the real side are as follows: $D = 6$, $D_{low} = 5$, $D_{high} = 12$, $D_{score} = 4$, $D_{out} = 2$. The degraded nodes for the real GossipSub are degraded on the application side, by making invalid publications and failing to relay messages.

The simulator is run on the same scale and on a single topic as well, with 20% of nodes degraded. The configuration uses the following parameters: $D = 8$, $D_{low} = 6$, $D_{high} = 15$, $D_{score} = 6$, $D_{out} = 2$. Scoring is also enabled. The degraded nodes in the simulator are configured to fail to relay correctly by making broken promises. This configuration is not an exact copy of libp2p’s internal parameters, but a profile calibrated to reproduce the observed topological properties as closely as possible. The aim of the validation is not for the internal configuration to be exactly the same, but rather for both configurations to result in the same topological behavior.

Snapshots of the real GossipSub are not reconstructed in the same way, as we do not have global access to the internal state of each node. The global graph is reconstructed from the **GRAFT** and **PRUNE** events provided by the extracted traces. As heartbeats are not synchronized in the real network, the snapshots are constructed in 1-second intervals, i.e. one heartbeat, as in the simulation snapshots. Since the time scale is the same on both sides, the data is more easily comparable. To prevent the start-up phase from skewing the results, measurements are taken over the last 20% of each time series.

The snapshots from the real GossipSub and the simulator take the same form so that the desired metrics can be easily extracted for both cases.

4.6.3 Analysis of the results

For the results below, on the simulator side, each network size was tested on three seeds. These seeds are the same as those used later for inference tests on the network: 24680, 13579, and 98765. The results presented are therefore an average. On the real network side, each run was executed once, to avoid monopolizing too many machines on the Grid5000 network for too long.

Table 4.2: Comparison of topological metrics: simulator vs real GossipSub

Size	Density	Mean degree	Components	Giant	Diameter	Mean path length	Clustering
20	0.309 / 0.305	5.87 / 5.80	5 / 5	16 / 16	2 / 2	1.511 / 1.517	0.329 / 0.389
50	0.123 / 0.146	6.03 / 7.16	11 / 11	40 / 40	3 / 4	1.979 / 1.918	0.140 / 0.201
100	0.061 / 0.069	6.08 / 6.86	21 / 21	80 / 80	4 / 4	2.351 / 2.328	0.070 / 0.107

The results show a good overall match between the simulator and the actual implementation. The discrepancies between the values are fairly small, and the quality improves as the network size increases, as GossipSub and its parameters are designed for networks where each node can choose its own neighbors.

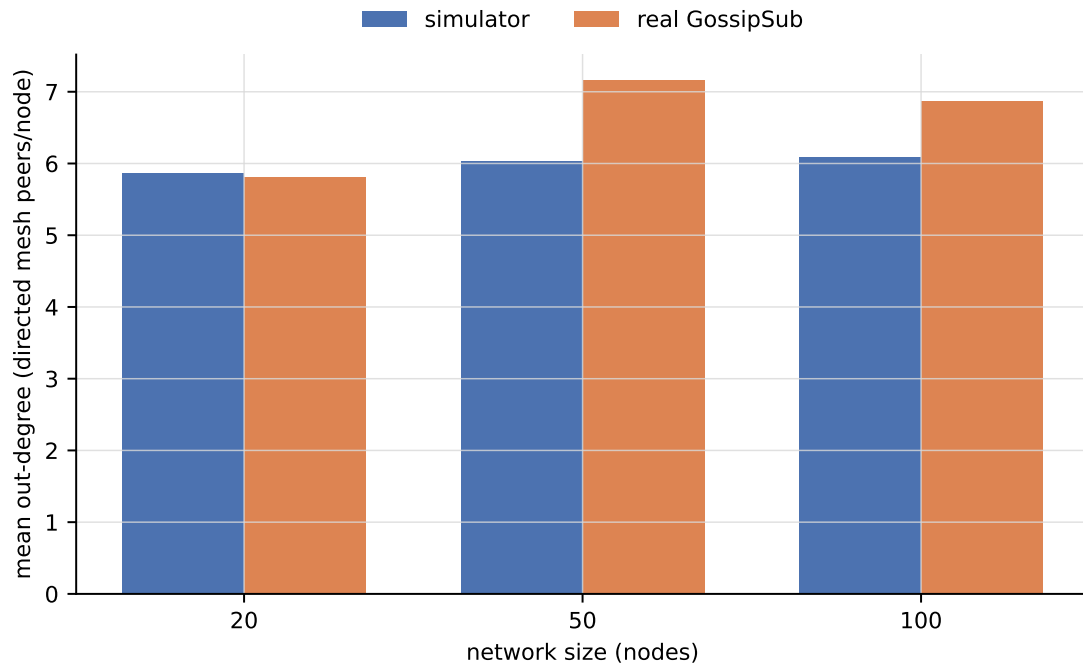


Figure 4.1: Average local degree by network size

The structural metrics show significant similarities. As shown in Figure 4.1, the average local degree of the mesh is similar between the simulator and the actual

run, with very similar results for 20 and 100 nodes and a slight difference for 50 nodes.

As shown in Figure 4.2 and Table 4.2, the other metrics confirm this similarity. The number of connected components is reproduced perfectly for all three sizes, as is the giant component. The diameter is also reproduced very well for 20 and 100 nodes. As for the average path length, it is very close, particularly for the 20 and 100-node size. Clustering, on the other hand, is the most unstable metric: it is underestimated at 20 nodes, 50 nodes and 100 nodes. Despite this discrepancy, the metrics are broadly aligned overall.

We can see that the exclusion of degraded peers is carried out in the same way in both cases and for the three network sizes. The giant is, in each case, equal to the network size minus 20% of it. The only nodes excluded from the overall graph are therefore those that behave poorly. When we see that there are 21 connected components for 100 nodes, this is because 20 nodes have been excluded due to their degraded behavior. There are therefore 20 isolated nodes and a connected component containing the remaining nodes. The same applies to the other two network sizes, scaling the figures accordingly.

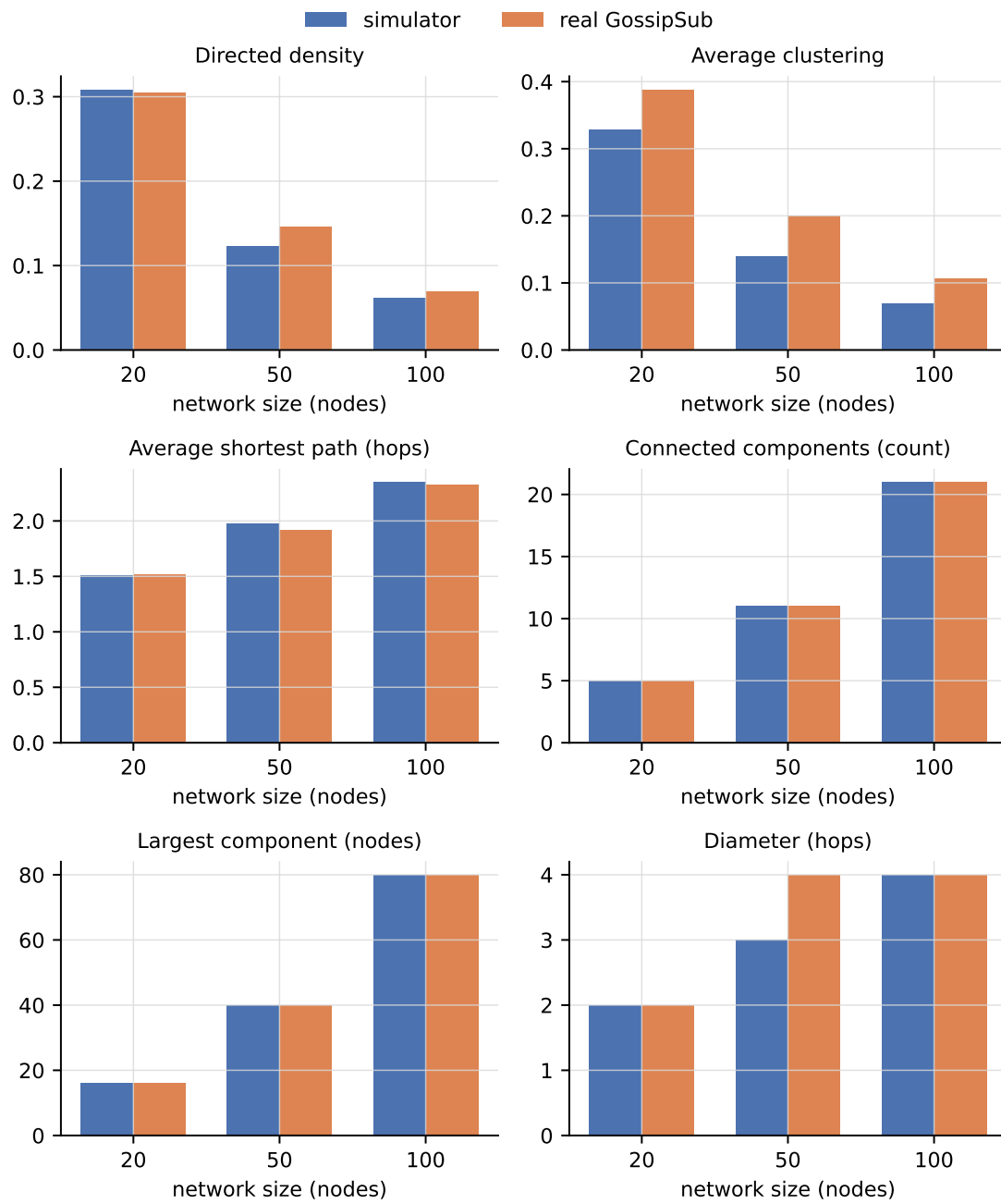


Figure 4.2: Global metrics by size

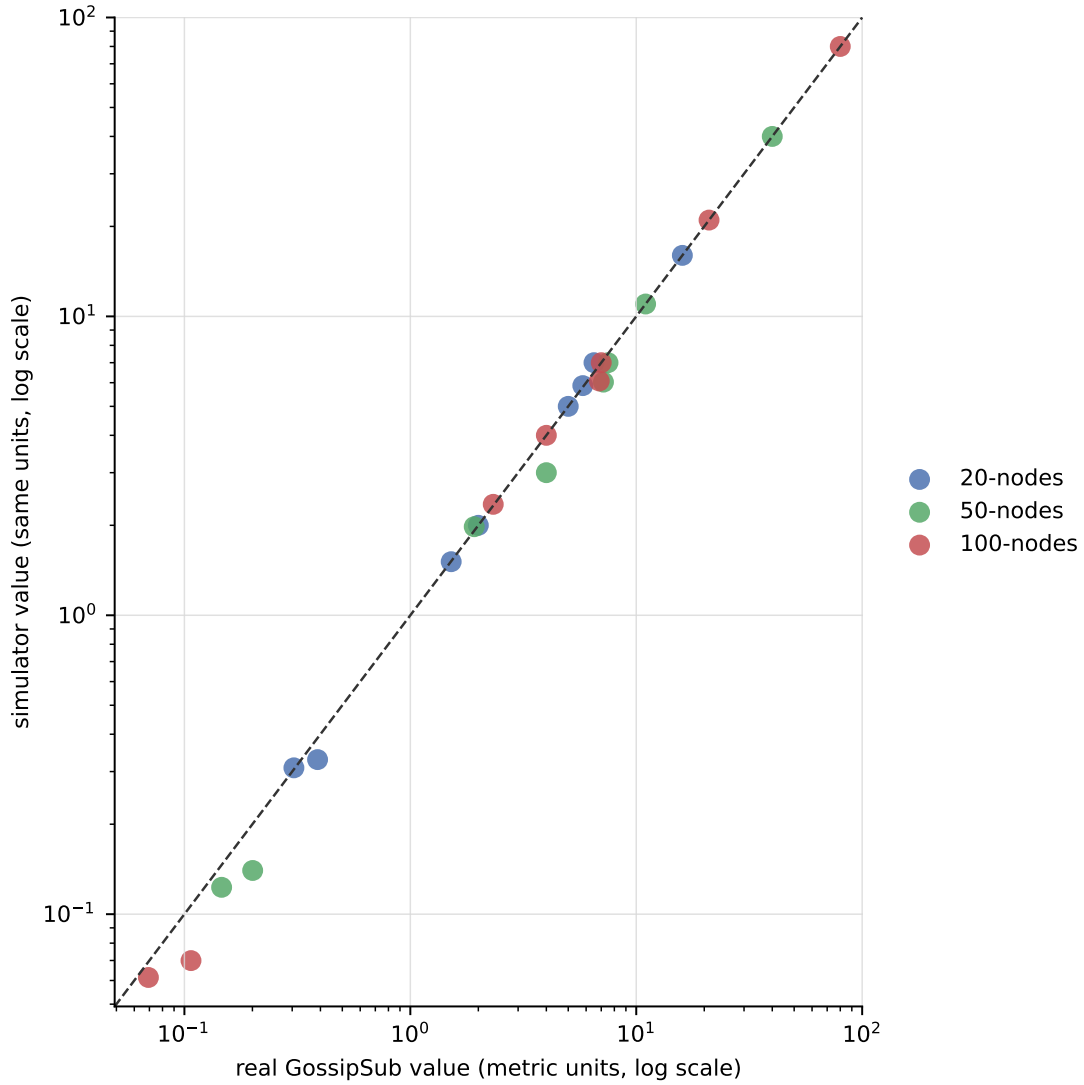


Figure 4.3: Metrics GossipSub VS Simulator

Figure 4.3 clearly shows that the metrics are well aligned around the ideal diagonal, particularly for 100 nodes. Following Figure 4.4, meanwhile, shows the ratio of the simulator's value to the actual GossipSub value for each network size. The dotted line at 1 represents perfect agreement, above 1, the metric is overestimated, and below 1, it is underestimated. We can see that most ratios are close to 1, indicating that the simulator correctly reproduces the structural behavior of the topology. Once again, at 100 nodes, the simulator is able to accurately reflect the properties of GossipSub due to its size.

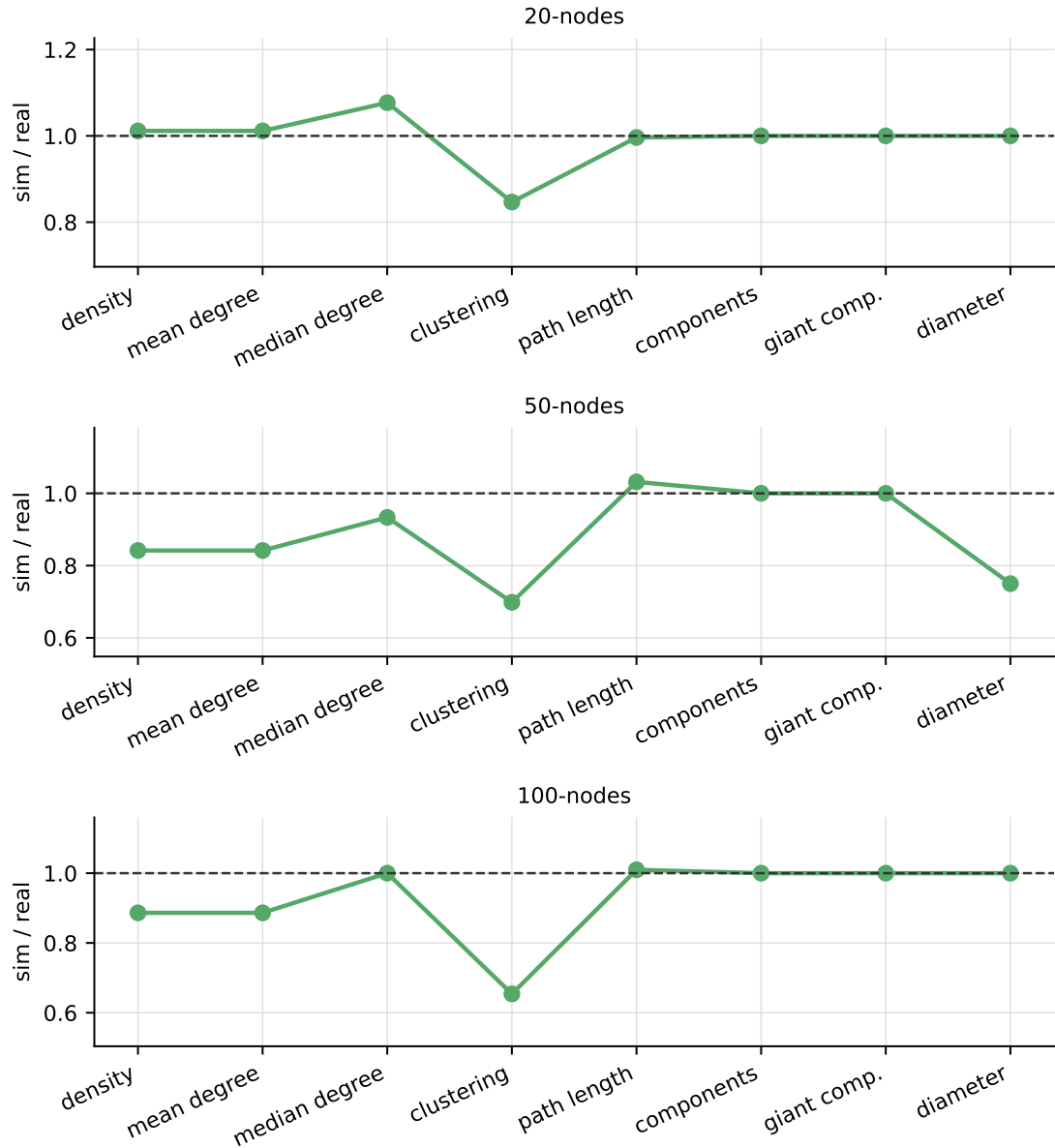


Figure 4.4: Metrics ratio by size

Taken together, these results show that the simulator correctly reproduces the geometry of the GossipSub mesh. Although it does not recreate exactly the same edges, it produces networks that exhibit the same structural properties.

Finally, following Figure 4.5 illustrates stability over time. We can see that both the simulator and the real-world implementation are fairly stable. The difference is that the simulator has downward spikes, as all the heartbeats occur simultaneously

in the simulator. So if every X heartbeats the mesh is re-evaluated to perform **PRUNE** or opportunistic **GRAFT**, in the simulator this will happen at exactly the same moment. In contrast, in a real-world implementation, each node has its own independent heartbeat, so no spikes are observed, the changes are correctly spread out over time. This distinction is not critical for our topological fidelity, and the simplification of the heartbeat was necessary to achieve a consistent time order in the simulator.

4.6.4 Conclusion

This validation demonstrates that the simulator satisfactorily reproduces the topological properties of GossipSub. This is particularly true for component structure, component size, diameter, average path length and degree distribution. The differences mainly concern temporal dynamics, as the simulator is more synchronized than a real implementation of GossipSub. The simulator is therefore suitable for studying topological trends, comparing inference strategies or scoring strategies, without claiming to reproduce the same edges at the same times.

Validation on networks with 20, 50 and 100 nodes provides reliable evidence to validate the simulator for larger networks, as the topological mechanisms being modeled are the same, the discrepancies are small, and the observed trends remain consistent as the network size increases. We will therefore carry out the inference tests on networks comprising 1,000 nodes in the simulator, for greater realism.

However, there are a few limitations to consider when using the simulator.

A significant limitation concerns how scoring is applied in the simulator. Degraded nodes are directly assigned a fixed penalty, without going through GossipSub’s behavioral mechanisms such as $P3$, $P4$, or $P7$. In a real-world deployment, these scores are built up gradually based on behavioral observations, which introduces detection latency and a risk of misclassification. In practical terms, this means that the isolation of degraded nodes in our experiments is likely overestimated: the simulator achieves a clearer separation than a real system would. Conclusions regarding the effectiveness of scoring as a countermeasure should therefore be interpreted as an upper bound rather than an accurate prediction. Once degraded nodes have been removed from the network, churn is very limited. However, the following experiments are short-term trials designed to demonstrate their feasibility. To assess the impact of churn on these experiments, it would be more useful to test them outside of a simulator over a longer period of time.

All links in the simulator share a fixed latency of 5 ms, which effectively neutralizes the $P2$ scoring component, designed to favor peers that deliver messages first. With uniform latency, no node has a structural advantage, and $P2$ does not discriminate between peers. In a real network, this component would help stabilize the mesh around low-latency peers. Its absence means that the observed topology

might be slightly less stable than in real-world conditions, and that the utility of *P2* cannot be assessed based on our experiments.

In the simulator, the 1,000 nodes execute their heartbeats at the same instant, whereas in a real implementation each node maintains its own independent timer. This artificial synchronization produces periodic spikes visible in the Jaccard similarity measures, as all opportunistic **GRAFTs** and **PRUNEs** are sent simultaneously. In a real network, these reconfigurations would be spread out over time. The impact is that temporal stability measures overestimate the amplitude of periodic disturbances.

The 1,000 nodes remain present from the start to the end of each simulation, with no nodes joining or leaving. This lack of churn facilitates inference attacks: a node identified through inference will always be an existing node, since no nodes leave the network. In a real network, a fraction of the discovered peers would eventually disconnect, reducing the precision of inference and potentially producing false positives. The precision and recall results for inferences therefore also constitute an upper bound.

Clustering is the topological metric that is reproduced slightly less accurately. This discrepancy can affect the results: in fact, during inference, the presence of more clustering can lower the results, because neighbors will provide results that their own neighbors have already provided.

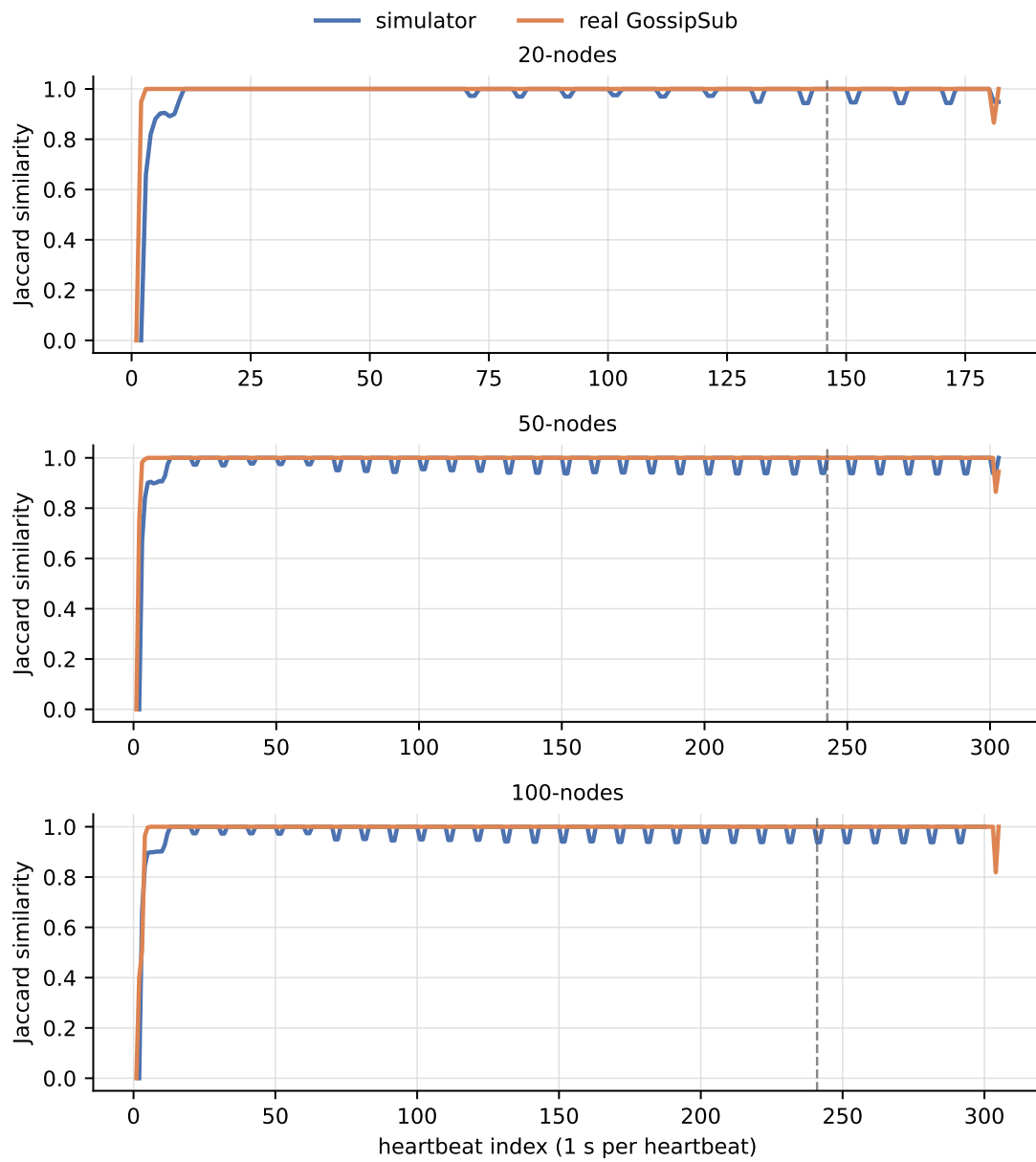


Figure 4.5: Jaccard timeseries by size

Chapter 5

Experiments and Data analysis

Using the simulator described in the previous chapter, experiments were conducted to test topology inference. These inference methods followed a step-by-step progression, with one method leading to the next. This chapter will detail the methodology as well as the various results obtained.

5.1 Evaluation methodology

To ensure that the experiments are conducted in a reproducible and controlled manner, the following sections will serve to define the framework and gain a better understanding of how the data is measured.

5.1.1 Scope of the inference methods

For the experiments, three different topological inference methods are being tested, each with a distinct objective.

The first method is the **passive ego mesh**. A single observer node typically participates in the network and passively observes what it sees locally. It does not attempt to position itself strategically, it only looks at its GossipSub neighborhood: its incoming and outgoing neighbors. The objective is to measure what a single node can infer from its ego mesh, its local view of the network. This answers the question: if I control a node, what can I learn?

The second method is the **passive coalition**. This method expands the scope of the first by forming a coalition of multiple observer nodes that observe passively at the same time. The nodes then combine their observations to pool their knowledge of the network. We test with different coalition sizes: 1, 2, 4, and 8 observers. The goal is to answer the following questions: What information gain can be achieved

when multiple observers collaborate? How much of the topology becomes visible if an adversary controls multiple nodes?

The third method is the **PX flood**. This is an active method, unlike the two previous methods, observer nodes do not merely collect signals, but trigger protocol responses to extract topological information. The principle is based on the peer exchange (PX) mechanism. When a node receives a **GRAFT** connection request in an already saturated mesh, it sends a **PRUNE** to refuse and attaches a PX list containing alternative nodes to connect to. This list contains peers that the node knows: it is either connected to them within its mesh or connected indirectly via gossip. This list therefore partially reveals the node’s neighborhood. Inference exploits this principle: a coalition of 4, 8, or 16 nodes simultaneously sends **GRAFTs** to a target node. This node’s mesh becomes artificially saturated, so it sends one or more **PRUNE + PX** messages. Each node in the list then becomes a target for the next wave, enabling recursive exploration and gradual network discovery. Inference thus proceeds in several waves. In each wave, the coalition nodes send a **GRAFT** to the target node. The target node accepts as long as it remains within the window $[D, D_{\text{high}}]$, and rejects the rest with **PRUNE + PX**. This principle can be applied recursively with a depth of one or two. Depth one examines the target node and its PX list, while depth two extends the exploration by one additional level. This method attempts to answer the following questions: to what extent can an active adversary map the GossipSub mesh by exploiting the peer exchange mechanism? What fraction of the network’s nodes can be discovered, and with what precision is the neighborhood of each target node reconstructed? Finally, to what extent does the attack disrupt the overall topology?

The global topology is determined by the evaluator after the inference and is never accessible to the attacker during the inference. We do not form coalitions larger than 16 nodes because having a large number of participating nodes comes at a cost: it is therefore not realistic for an attacker to have too many nodes, the goal is to be able to perform inference with a small number of nodes in order to keep costs down.

Each of the three inference methods is evaluated across its full range of parameters. For example, for **passive coalition**, inference is evaluated for 1, 2, 4, and 8.

5.1.2 Scenarios

The GossipSub simulator allows users to create configuration files that define a scenario: an experimental framework. Within these scenarios, users can modify GossipSub’s behavior, the behavior of the nodes, and the activation or deactivation of certain protocol mechanisms. Two scenarios are examined for each method of inference:

The first scenario is **Baseline Without Scoring**. This scenario serves as a control. In this case, 20% of the nodes are degraded, but GossipSub’s scoring is disabled. The degraded nodes can therefore exhibit problematic behavior but are not excluded from the network. The degraded nodes make broken promises: they send **IHAVE** messages but do not send the message when a node responds to them with **IWANT**. This scenario allows us to observe what happens when nodes are degraded but remain in the network. It serves as a point of comparison with scenario where scoring is enabled, to measure the difference in difficulty this creates for inference. The degree parameters are: $D = 6$, $D_low = 4$, $D_high = 8$, $D_score = 4$, $D_out = 2$.

The second scenario is **Validated Scoring**. This is the main and most important scenario. In fact, it was calibrated based on a comparison between the actual implementation of GossipSub on Grid5000 and the simulator. Unlike the **baseline** scenarios, degraded nodes do not simply ignore **IWANTs**, they also do not relay messages received in the mesh. The degree parameters are also different: $D = 8$, $D_low = 6$, $D_high = 15$, $D_score = 6$, $D_out = 2$. The scoring changes the weights of the various elements and penalizes broken promises much more heavily, with a significant penalty for degraded peers. This scenario represents the profile that most closely matches the real GossipSub experimentally. It serves as a basis for comparison with the other scenarios and as a point of validation against the real GossipSub.

As explained in the section 5.1.1, each inference method is evaluated across all combinations of its parameters. Each experimental configuration is then run on the two scenarios presented above. Finally, to limit the dependence of the results on a favorable seed, each run is repeated with three seeds: 24680, 13579, and 98765. The attacks described in this chapter are therefore reproducible.

Inferences and measurements always begin after the simulator’s 10th heartbeat, because the first 10 heartbeats are warm-up heartbeats that allow the network to form properly. The two scenarios are run on a network of 1,000 nodes. The two scenarios are being tested in a single-topic setting.

5.1.3 Sources of topological information leakage

The topology inference methods studied are based on a key principle: even though GossipSub does not expose its topology, maintenance and propagation messages locally reveal parts of the topology. An observer within the network can exploit these signals to attempt to reconstruct part of the graph.

In our models, the inferences do not use the simulator’s internal state. Observer nodes simply use the local information relevant to them, relying primarily on the **GRAFT**, **PRUNE**, **IHAVE**, **IWANT**, and GossipSub messages they receive.

The most directly topological signals are **GRAFT** and **PRUNE**. When a node sends

or receives a **GRAFT** with a peer, this establishes a bidirectional link in the mesh, with both nodes adding each other to their respective meshes. Similarly, a **PRUNE** exchanged between two nodes removes this link on both sides. For simplicity, we therefore treat these relationships as undirected in our inference: a **GRAFT** observed in either direction adds an edge between the observer and the peer, and a **PRUNE** observed in either direction removes it. Strictly speaking, mesh membership may be asymmetric at certain transient moments, for example, in the brief window between when a node sends a **GRAFT** and when its peer processes it, but these cases are sufficiently rare and fleeting for the undirected approximation to remain valid in practice.

The other messages, **IHAVE**, **IWANT**, and data messages, do not provide direct proof of topology, they do not create edges. However, they can indicate a certain proximity for a given topic with an observer. The observer can therefore retain this information to combine it with other data later.

Finally, the peer exchange (**PX**) list provides direct information about the topology, since it is constructed using peers that the node creating it knows or is directly connected to.

5.1.4 Architecture of inference methods

Each inference method followed a specific architecture for its implementation, the following section provides details.

For **passive ego mesh**, the observer maintains a set: neighbors. As described in section 5.1.3, a sent **GRAFT** adds an outgoing edge from the observer to the peer. A received **GRAFT** adds an incoming edge from the peer to the observer. **PRUNE** messages follow the same behavior. **IHAVE**, **IWANT**, and data messages are recorded as activity proofs but do not create edges. In this inference and the following ones, we do not consider the direction of the edges, only the existence of an edge between two nodes, because there are very few situations in GossipSub where a mesh relationship is not bidirectional. It is therefore more advantageous for an inference to assume that an edge works in both directions.

Regarding **passive coalition**, each coalition member maintains a sets: neighbors. The coalition then constructs the union of these views. An edge is added to the inferred topology if at least one coalition member has observed it. It is thus a combination of all viewpoints, a multiplication of the principles of the first inference.

The **PX flood** method introduces a depth and wave dimension. The adversary does not simply leave its nodes in the same location. The inference begins with the coalition targeting a node. Each node in the coalition sends a **GRAFT** to the target node of the inference. The goal of the operation is to reach the node limit in the mesh so that the **GRAFT**s are rejected and the target node responds with

a PRUNE + PX. If multiple nodes obtain a PX list for the target node, that's even better. We will traverse the PX list and extract all the nodes found there, and we will repeat the same flooding operation on them to obtain their PX lists. Figure 5.1 illustrates the inference process. Whether the depth is 1 or 2, we begin the inference on node $T0$ during wave 0. Its PX list gives us two nodes: $P1$ and $P2$. During waves 1 and 2, we repeat the operation on $P1$ and $P2$ to obtain their PX lists. At depth 1, we stop here, at depth 2, we go one level deeper by repeating the operation on the nodes obtained from the PX lists of $P1$ and $P2$ during waves 3, 4, 5, and 6. This is only a schematic example, in reality, a PX list can contain up to 16 elements, so there are many more waves per depth if the operation is successful. A wave is launched every two heartbeats.

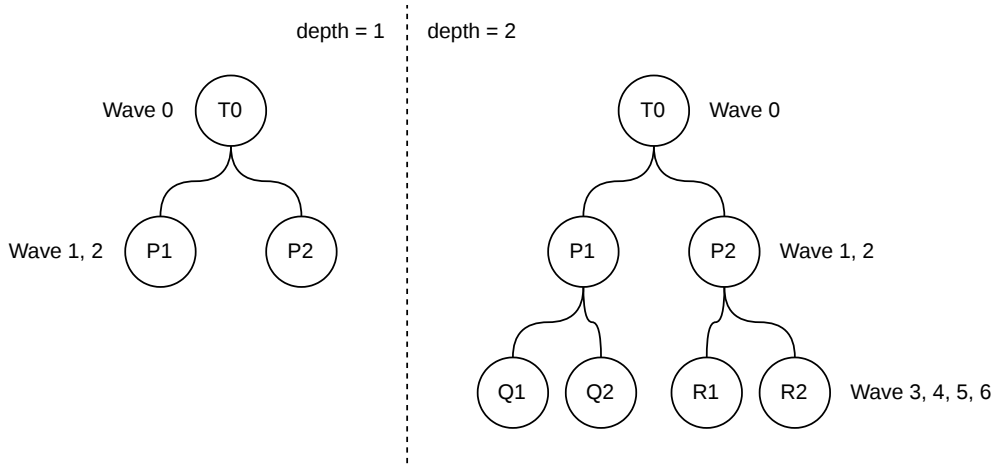


Figure 5.1: PX flood - depths and waves

5.1.5 Metrics

To evaluate the different inference methods, we rely on standard set-comparison metrics as well as coverage metrics. In all cases, the inferred topology is compared to the topological ground truth extracted by the simulator. Depending on the context of the inference, the comparison may involve the local ego mesh, the coalition boundary, or the entire network.

Precision, recall, and F1-score. Let E_{pred} denote the set of inferred edges/nodes and E_{true} the corresponding ground-truth edge/node set. From these two sets, we define:

- TP (true positives) as the number of inferred edges/nodes that actually exist in the ground truth,
- FP (false positives) as the number of inferred edges/nodes that do not exist in the ground truth,
- FN (false negatives) as the number of true edges/nodes that were not recovered by the inference.

Precision measures how many of the predicted edges/nodes are correct:

$$\text{Precision} = \frac{TP}{TP + FP} \quad (5.1)$$

Recall measures the extent to which the true topology is recovered:

$$\text{Recall} = \frac{TP}{TP + FN} \quad (5.2)$$

F1 combines precision and recall to produce a balanced metric:

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (5.3)$$

When calculating precision, recall, or F1 on edges, it is always done on undirected edges.

Jaccard similarity. For two sets A and B , the Jaccard similarity is defined as:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \quad (5.4)$$

This measure allows us to assess the similarity between two sets: the result ranges from 0 (no elements in common) to 1 (identical sets). Here, it is used to compare two snapshots of the topology: a value close to 1 indicates that the topologies at the two points in time are nearly identical, while a value close to 0 means that they have diverged significantly.

Global edge coverage. For coalition-based inferences, we measure how much of the true global mesh is covered by the inferred edge set:

$$\text{GlobalEdgeCoverage} = \frac{|E_{\text{pred-edges}} \cap E_{\text{global-edges}}|}{|E_{\text{global-edges}}|} \quad (5.5)$$

Here, $E_{\text{pred-edges}}$ is the set of edges inferred by the inference, and $E_{\text{global-edges}}$ is the set of all true mesh edges in the global topology.

Global node coverage. For coalition-based inferences, we measure how much of the true global mesh is covered by the inferred node set:

$$\text{GlobalNodeCoverage} = \frac{|E_{\text{pred-nodes}} \cap E_{\text{global-nodes}}|}{|E_{\text{global-nodes}}|} \quad (5.6)$$

Here, $E_{\text{pred-nodes}}$ is the set of nodes inferred by the inference, and $E_{\text{global-nodes}}$ is the set of all true mesh nodes in the global topology. Global node coverage is calculated based on the peers in the final mesh, since degraded peers are excluded from the network, they are not included in the coverage calculation.

Drift. For active inferences, we measure a perturbation of the network based on the directed edges. To do this, we compare the network before inference with the network after inference. This drift is the Jaccard distance ($1 - \text{Jaccard}$) between the reference set of directed edges and the set of directed edges after inference:

$$\text{drift} = 1 - \frac{|E_{\text{ref}} \cap E_{\text{inf}}|}{|E_{\text{ref}} \cup E_{\text{inf}}|} \quad (5.7)$$

Here, E_{ref} is the reference set for the heartbeat immediately before active inference, and E_{inf} the set with inference.

High degree recall. The proportion of the top 20% most connected nodes that have been discovered. Calculated based on undirected edges.

Low degree recall. The same applies to the remaining 80% of nodes.

Wave progression. Unique cumulative count of PX pairs identified by wave number

Convergence time. Number of heartbeats required to reach the 50%, 90%, 95%, and 100% thresholds for final neighborhood coverage.

5.2 Results for passive ego-mesh inference

This method serves as the first building block for the rest of the inferences. Before performing large-scale inferences, it is necessary to determine whether a node-level inference is possible. In this case, a single passive observer attempts to reconstruct its local mesh. The question this method seeks to answer is: how much of the local neighborhood can a node reconstruct using only local signals?

In Figure 5.2 we can see that the ego-mesh can be reconstructed with sufficient accuracy for our purposes. For all scenarios, the reconstruction score is perfect.

This first building block will then be useful for reconstructing a larger topology by combining the views of multiple nodes because we can be sure that a node can easily see its neighbors.

In Figure 5.3 is a visual representation of a node attempting to infer its mesh using the **validated scoring** scenario. Unsurprisingly, it finds all the connections and thus all its neighbors, as the previous results show.

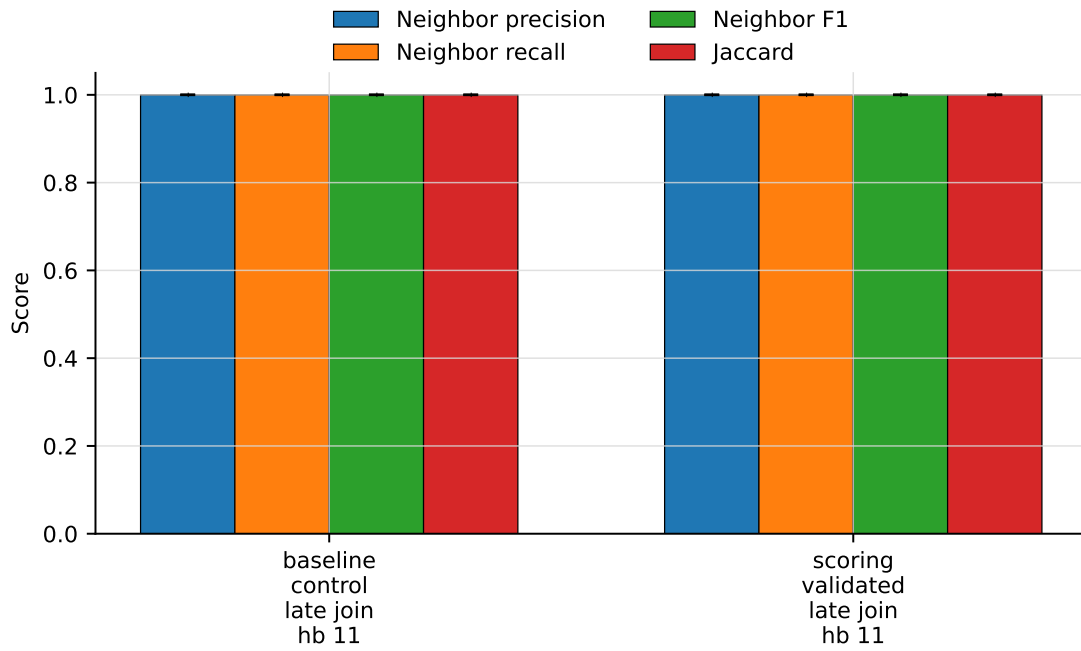


Figure 5.2: Passive ego mesh - metrics by scenario (mean across seeds)

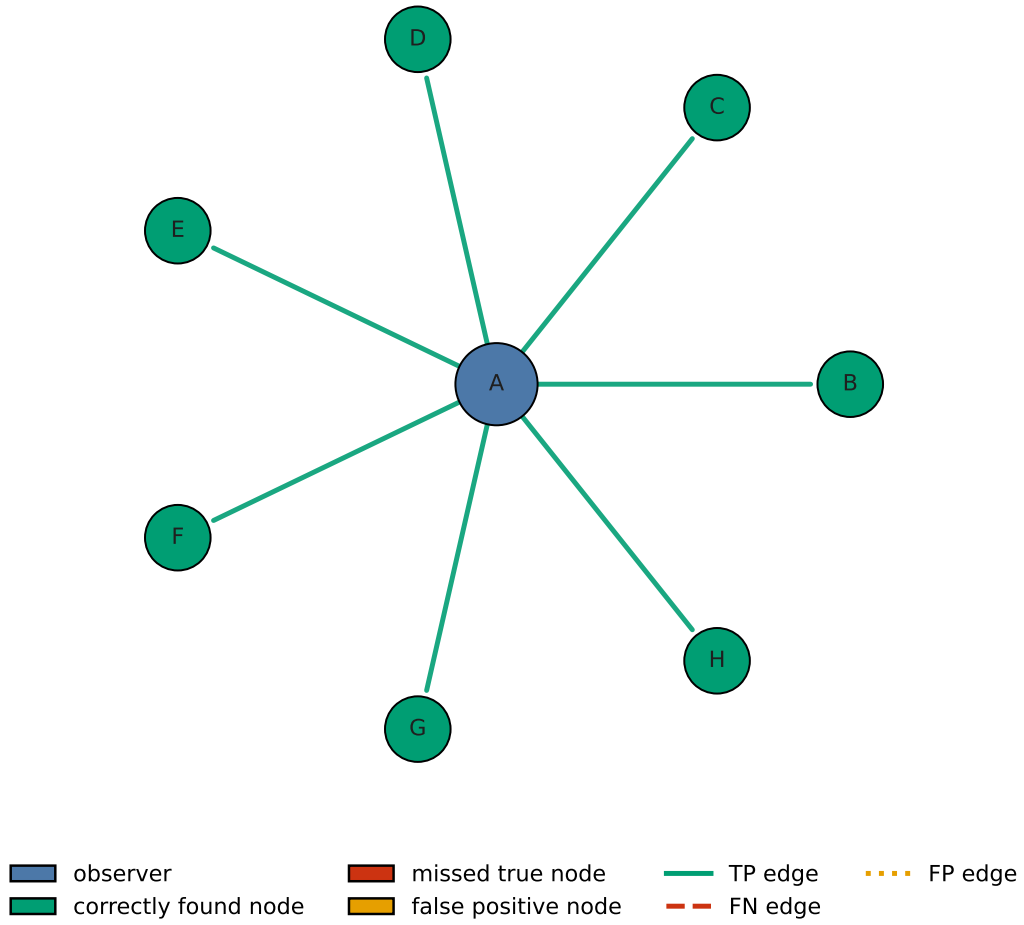


Figure 5.3: Passive ego mesh - inference

It is interesting to note that the observer managed to reconstruct the entire mesh, but we must consider the time it takes to do so. For if an inference precedes an attack, it is important that the inference be fast so that the attack can be launched quickly. If, for example, we want to block information spreading through the network, time may be of the essence. Speed is therefore a key element of inference.

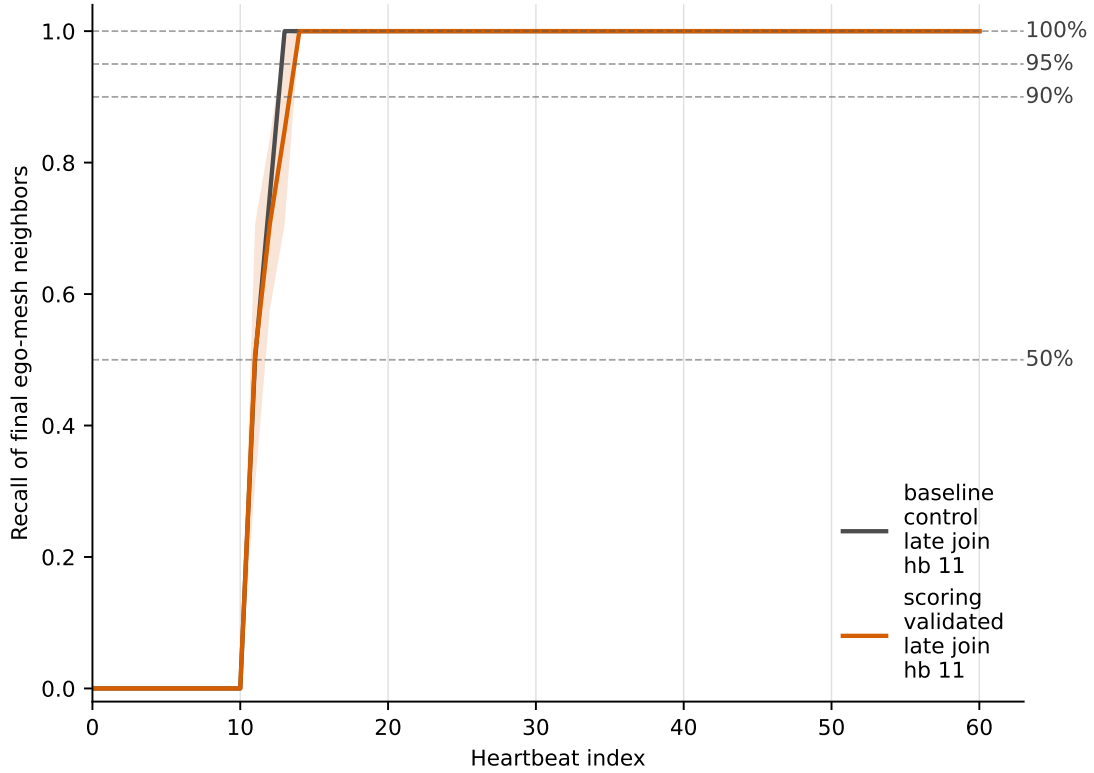


Figure 5.4: Passive ego mesh - time to final neighbor knowledge (mean across seeds)

Table 5.1: Number of heartbeats per scenario and threshold (mean $\pm$ std)

Scenario	50%	90%	95%	100%
Baseline	11.00 $\pm$ 0.00 hb	13.00 $\pm$ 0.00 hb	13.00 $\pm$ 0.00 hb	13.00 $\pm$ 0.00 hb
Validated	11.33 $\pm$ 0.47 hb	13.67 $\pm$ 0.47 hb	13.67 $\pm$ 0.47 hb	13.67 $\pm$ 0.47 hb

We can see on Figure 5.4 and Table 5.1 that inference occurs very quickly to reach the various knowledge thresholds. The node joins the network at heartbeat 11, so inference begins at heartbeat 11, as the first 10 heartbeats are used for network warmup. All scenarios take roughly the same amount of time to reconstruct the entire mesh, except for **validated scoring**, which takes a little longer, but this additional time is negligible. They all take a maximum of about 3 heartbeats to reach 100% coverage of their mesh.

5.3 Results for passive coalition inference

This second inference aims to combine the local perspectives of several passive nodes in the network to reconstruct a global view. This inference has several objectives: to correctly reconstruct the observable boundary of the set of nodes involved in the inference, and to measure the portion of the global network that this boundary allows to be discovered. And finally, to examine the extent to which increasing the number of involved nodes improves discovery. The inference also begins at heartbeat 11 after the warmup.

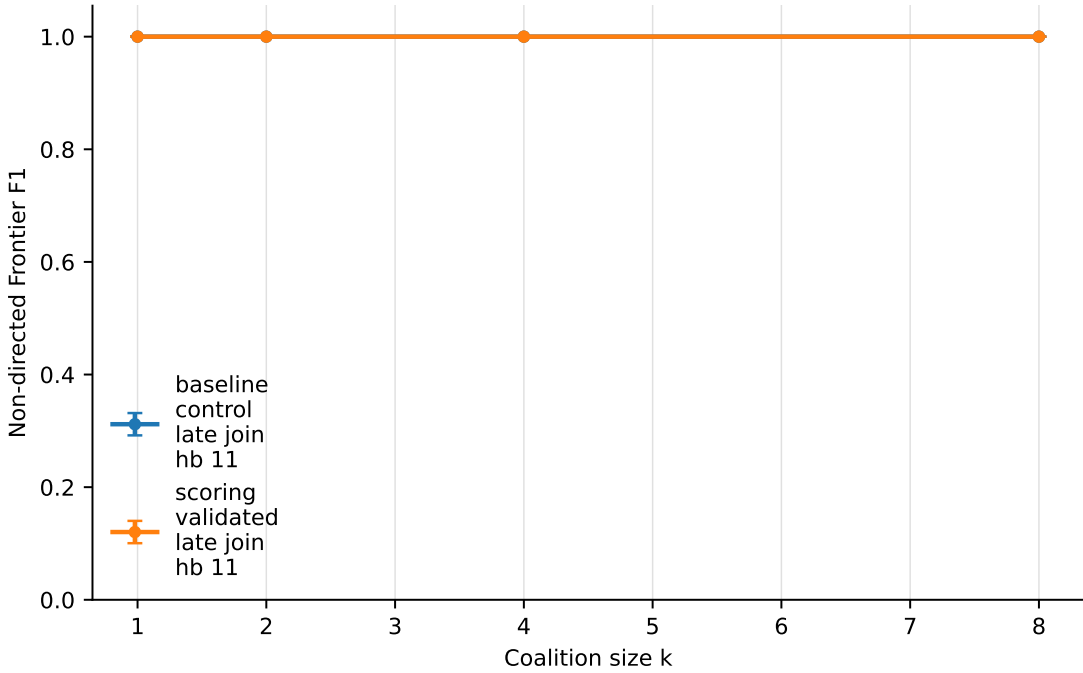


Figure 5.5: Passive coalition - frontier reconstruction F1 (mean across seeds)

Figure 5.5 follows the logic of the results from the passive ego mesh: whether there are 1, 2, 4, or 8 nodes, they all fully recover their observable boundary. Since a single node fully recovers its boundary, a combination of multiple nodes also fully recovers their combined observable boundary.

Now that we know the coalition has reached its maximum discovery potential, to what extent does increasing the size of the coalition help in discovering the overall topology?

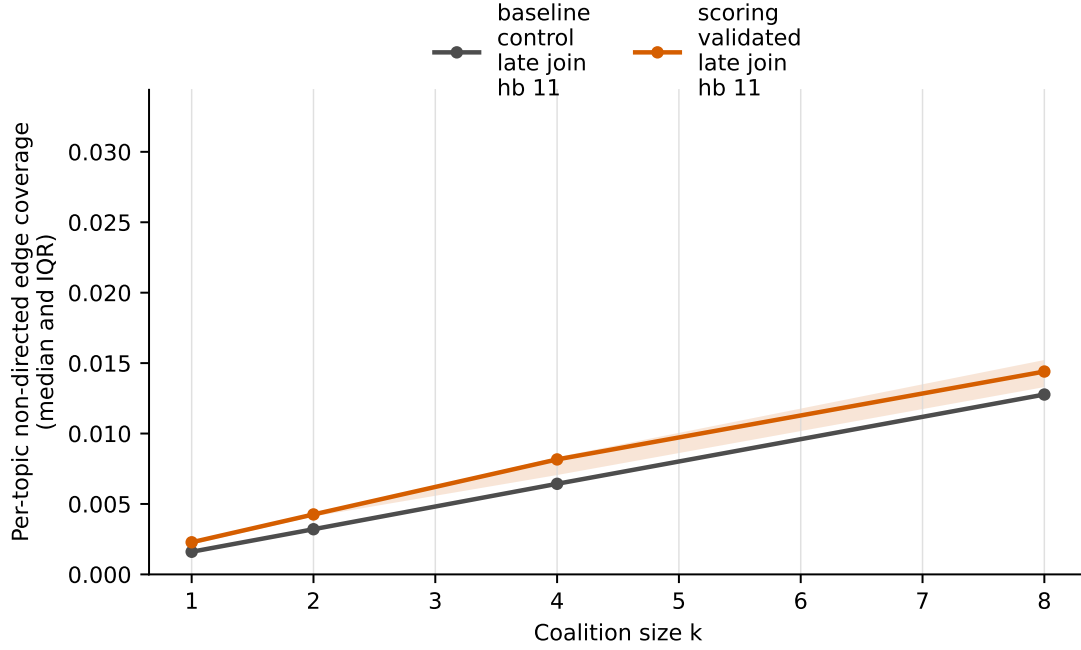


Figure 5.6: Passive coalition - global edge coverage (median across seeds)

We can see here on figure 5.6 that adding nodes does indeed increase overall coverage, but only linearly. Since increasing the number of observer nodes in a P2P network can be costly, we need to find a smart way to optimize each node's view. Determining the complete set of nodes in a network does not seem realistic at first glance. It's also important to realize that while we're discovering more than 0.01% of the network's edges here, these aren't actually the edges we're interested in. In fact, our current inference technique only allows us to identify edges that connect to one of the nodes in our coalition. We currently have no way of inferring the edges between two nodes outside the coalition. So for now, we should focus on the number of nodes we find, we cannot yet determine the shape of the topology, but at least we can identify the nodes that make it up.

In Figure 5.7, we can see that the time required to find the entire neighborhood of the coalition is roughly similar to the time shown in Figure 5.4 for a single node. All nodes find their neighborhoods in parallel at the same speed. The speed therefore does not change between a single node and a coalition of multiple nodes.

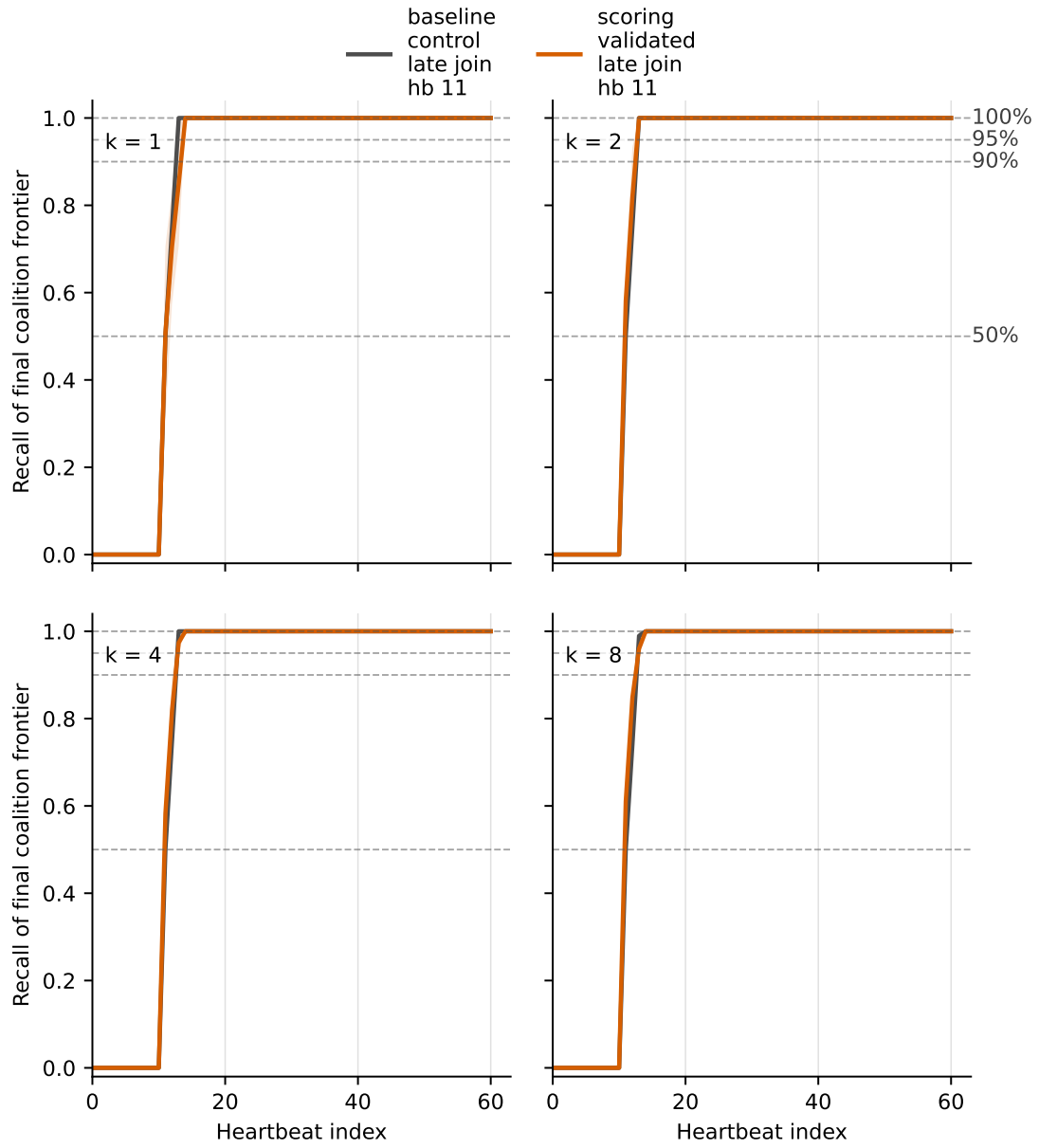


Figure 5.7: Passive coalition - time to final neighbor knowledge (mean across seeds)

Since the edges are only those that connect to the nodes in our coalition, let's focus instead on the coverage of the nodes we identify through this inference. In Figure 5.8, we can see that the scenario achieving the best node coverage is **scoring validated**. We achieve over 6 percent coverage with 8 nodes on our 1,000-node network.

$$degree \times coalition_size = number_of_nodes$$

$$\frac{number_of_nodes}{1000} \times 100 = upper_bound_coverage\%$$

$$8 \times 8 = 64 \quad \frac{64}{1000} \times 100 = 6.4\%$$

$$6 \times 8 = 48 \quad \frac{48}{1000} \times 100 = 4.8\%$$

Here are the ideal percentages that could be achieved with an 8-party coalition in both cases. Obviously, the parameter D isn't always met, sometimes it can be slightly lower, for example, which is why we don't reach these exact figures but fall slightly short of them.

This figure makes sense given that each additional node contributes its degree to the overall knowledge of nodes. The other scenario have lower and relatively similar node coverage, the target degree D for the **baseline** scenarios is lower: 6 compared to 8 for **scoring validated**. This leads to a weaker overall understanding. The problem is that the evolution is linear, so it is not possible to rely on the evolution of the coalition size to identify the nodes in the network.

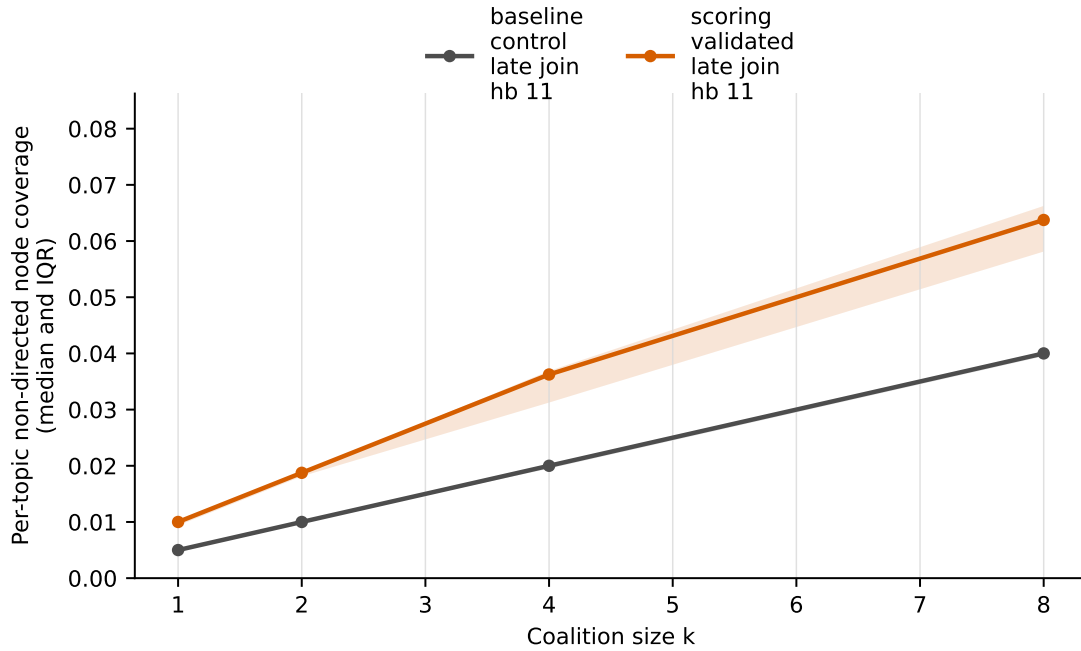


Figure 5.8: Passive coalition - global node coverage (median across seeds)

5.4 Results for PX flood

5.4.1 Global coverage

Unlike the two previous inference methods, **PX flood** inference is an active inference method. Rather than observing the traffic passing through its coalition, the coalition of nodes simultaneously sends **GRAFT** messages to a target to saturate its mesh. Once the mesh is saturated, the target node sends **GRAFT** rejection **PRUNE** messages containing a peer exchange list. Each **PX** list reveals a sample of the peers connected to the target, and these peers become the next target for the next wave. The process repeats recursively until all proposed nodes have been explored or the simulation stops.

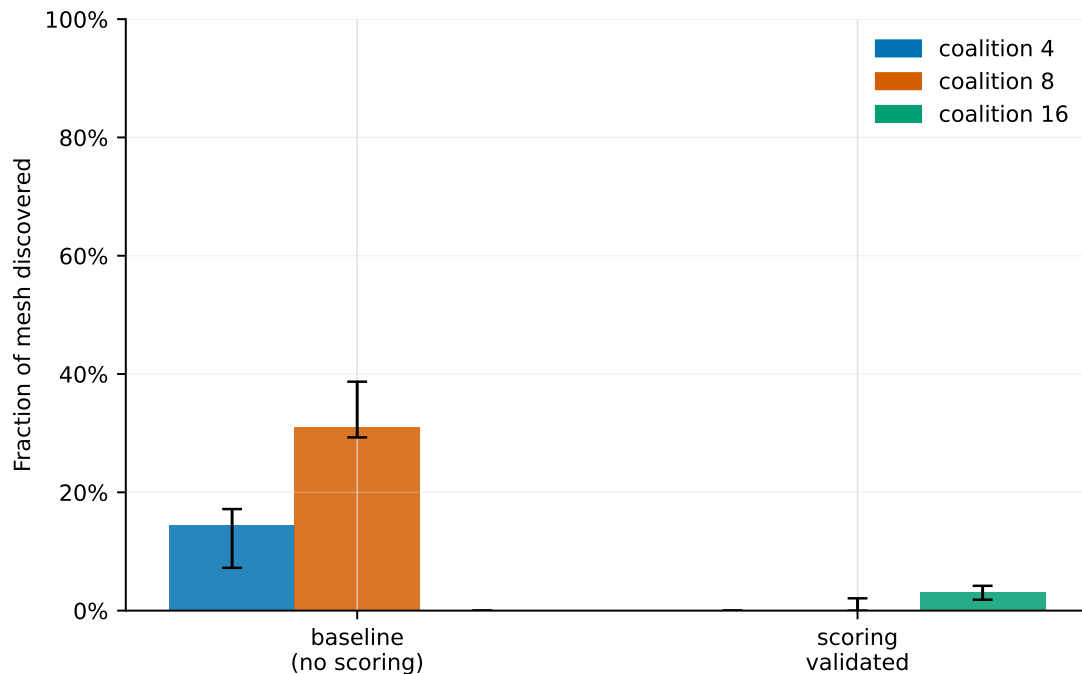


Figure 5.9: PX flood - coverage depth 1 (median across seeds)

Figure 5.9 shows the fraction of the mesh discovered as a function of the scenario and the coalition size, with a wave depth of 1. Coalition size is the factor that most significantly affects coverage. In the **baseline** scenario, a coalition of 4 uncovers 14.5% of the network nodes, while a coalition of 8 uncovers up to 31.0%. This is more than twice the coverage of the coalition of 4, the relationship is therefore not linear, as a larger coalition results in more **PRUNE** operations per wave, each with a distinct **PX** list.

The **validated scoring** scenario is resistant to attacks thanks to its structure. With coalition size set to 4 and 8, no nodes are exposed during the run. This is because D is 8 and D_high is 15. This means the node is highly likely to accept all **GRAFTs** from a coalition and not send any **PRUNE + PX**. When coalition size is 16, we are certain to exceed the D_high threshold and receive **PRUNE** messages. Here, the results suggest that we tested a target that performed poorly and was gradually excluded from the network during the warmup period. The 16 nodes thus filled the node’s mesh, and only the 16th node received a **PRUNE + PX**. Given the limited information source, a single **PX**, the discovery rate is only 3.2%.

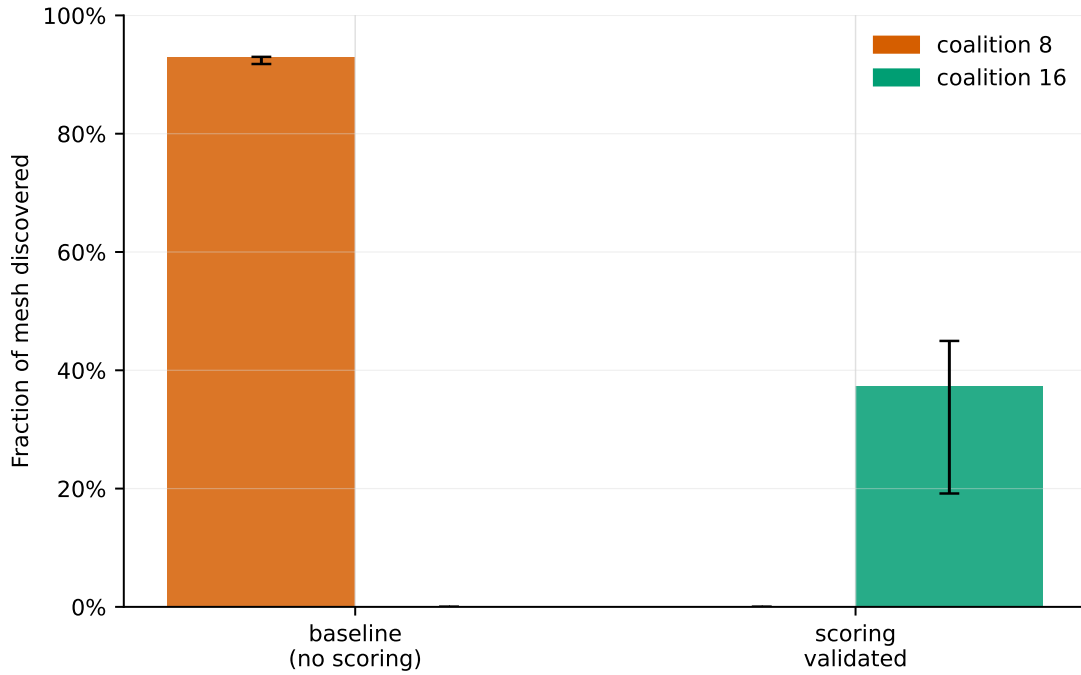


Figure 5.10: PX flood - coverage depth 2 (median across seeds)

As shown in Figure 5.10 and the corresponding data in Table 5.2, increasing the wave depth from 1 to 2 results in a very large increase. For the **baseline** scenario, the increase is 31% to 92.8%.

This significant increase is expected and is fully consistent with the depth principle. When the depth is increased by one, active inference is re-run on every node in each of the 16 **PX** lists already discovered at the previous depth reached. There is therefore enormous potential for improvement.

In the **validated scoring** scenario, the gain associated with depth is also high. With 16 nodes, the rate increases from 3.2% to 37.2%. The explanation lies in the poor start of depth 1, since little has been explored yet and there are 16

nodes to tackle, the discovery potential is high. These targets are standard targets, which require fewer nodes to fill their mesh, therefore, each target generates a large number of PX lists.

Table 5.2: PX Flood : discovery coverage and topological disruption (median across 3 seeds)

Scenario	cs ¹	Depth	Coverage	Final drift
Baseline	4	1	14.5% (0–20%, high variance)	1.1%
Baseline	8	1	31.0%	4.1%
Baseline	8	2	92.8%	33.0%
Validated scoring	4	1	0%	4.9%
Validated scoring	8	1	0%	4.9%
Validated scoring	16	1	3.2%	6.2%
Validated scoring	16	2	37.2% (1–53%, high variance)	9.9% (6–14%)
Validated scoring	16	3	10–99% (high variance)	6–45% (high variance)

The detection precision is 100% in all configurations.

5.4.2 Neighborhood Reconstruction by Target

We have good overall coverage, but how well do we perform at correctly reconstructing the mesh of attacked nodes based on their PX list? We will now focus on the average micro-precision and micro-recall across the evaluated targets. Micro-precision and micro-recall refer to precision and recall at the level of a node’s mesh, rather than at the level of the entire network.

¹coalition size.

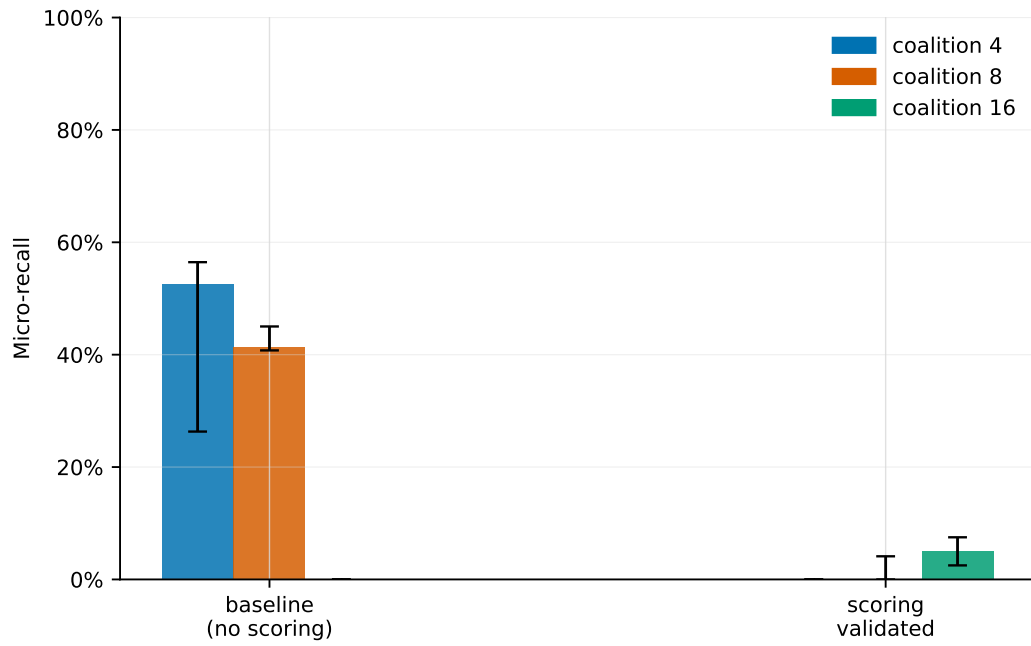


Figure 5.11: PX flood - micro recall (median across seeds)

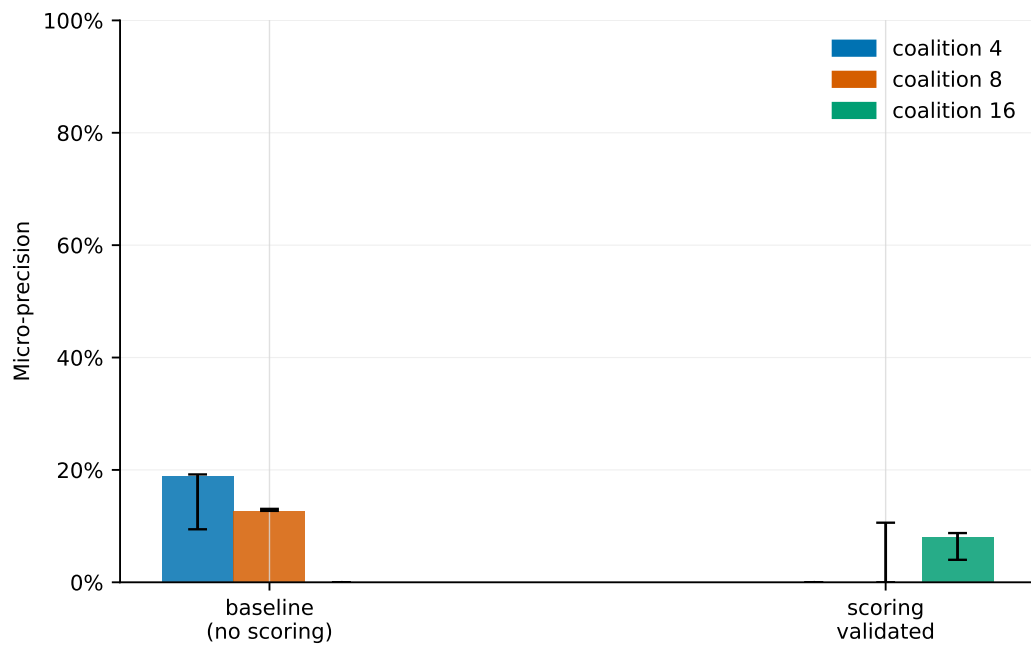


Figure 5.12: PX flood - micro precision (median across seeds)

Figure 5.11 shows that micro-recall is 41% for coalition size set to 8. This is the case for **baseline** scenario, confirming that the inference recovers more than half of the true neighbors of each target. In contrast, Figure 5.12 shows that micro-precision is much lower: 13% in **baseline** scenario for a coalition of 8 nodes.

This asymmetry does not indicate a failure of inference but is inherent to the characteristics of PX in GossipSub. The selection of nodes to be included in a PX list is made from among all peers connected to the node generating the list. Among these connected nodes are the mesh nodes, but not exclusively, there may also be nodes with which gossip is exchanged, or nodes discovered via an external discovery system. The list can contain up to 16 elements, whereas in the standard GossipSub configuration, D is 6. By definition, if the list is full, there will be 10 nodes that are not part of the mesh. Precision is therefore bounded by the ratio of the mesh size to the size of the PX list.

For the true GossipSub:

$$\frac{6}{16} \approx 38\%$$

For example, in the case of the simulator for the validated scenario:

$$\frac{8}{16} = 50\%$$

And these are best-case percentages, when we aggregate the results of multiple PX runs performed on the same node, the precision decreases.

As for overall precision, that is, the probability that an inferred node is actually present in the overall network, this precision is 100% in all configurations. The inference never produces false positives: every inferred node exists in the network.

5.4.3 Wave progression

Figure 5.13 shows the number of pairs discovered in the PX lists as a function of wave number, by scenario and coalition size. When the coalition consists of 4 nodes, the attack's potential is quickly exhausted. A substantial number of new nodes are discovered at the beginning, but eventually a plateau is reached. The inference stops even before the end of the simulation. The plateau is explained by the fact that the further we go, the more we discover nodes that we already know from previous waves. This property is called clustering: neighbors share many common peers. The maximum we could theoretically achieve in a single wave is the size of the coalition multiplied by 16, the size of the PX list: 128 for a coalition of 8. But in practice, we never reach this number because only a subset of the coalition's nodes receive a PX list. And even if we had 8 PX lists for a coalition of 8 nodes, redundancy would prevent us from reaching the theoretical maximum. We see that

two cases, the simulation stops before inference is complete, which means that our current results are partly limited by the simulation time.

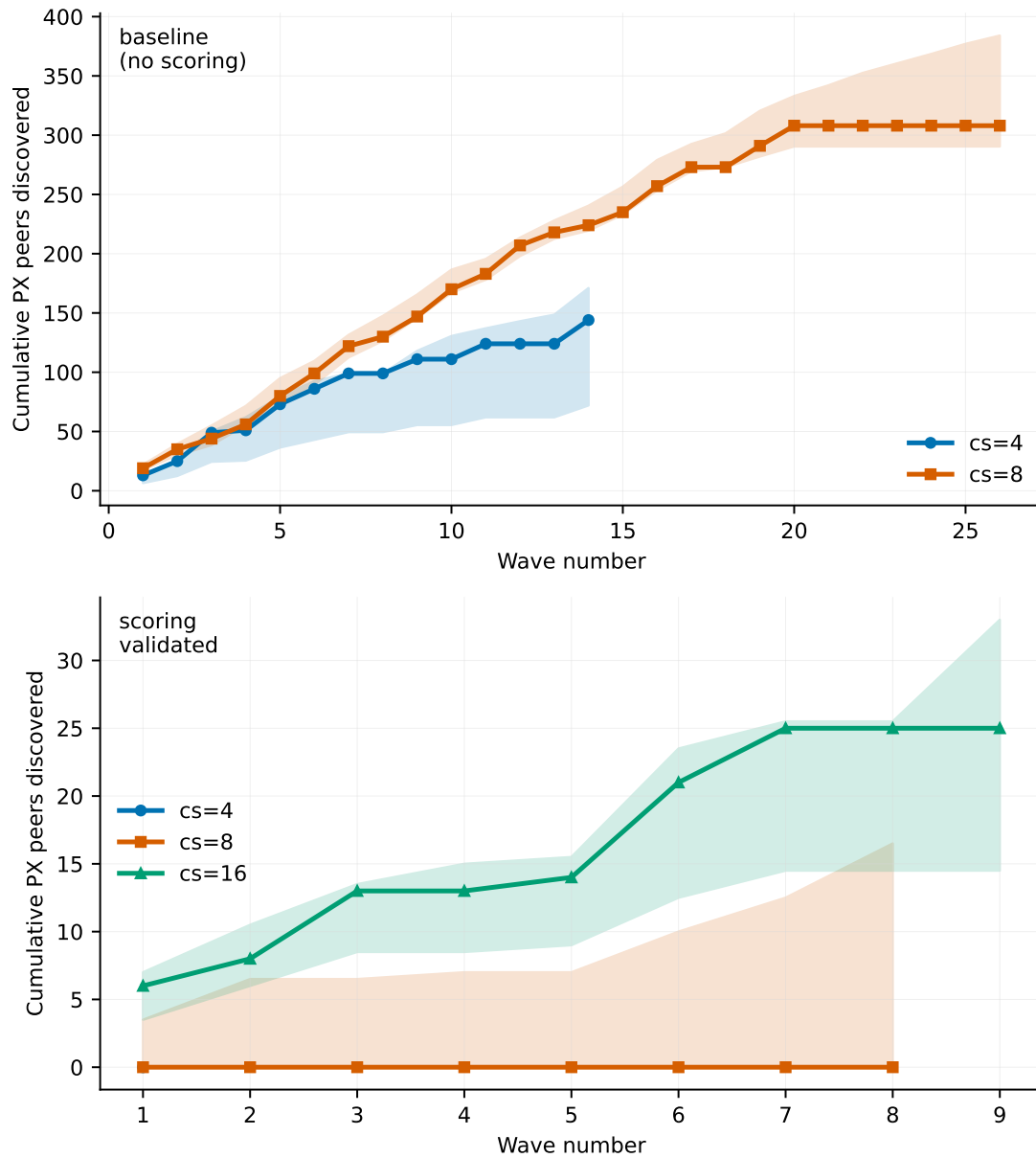


Figure 5.13: PX flood - wave progression (median across seeds)

Since we observed that simulation time limits the potential results, to overcome this limitation, we tested a depth of three on one scenarios: **validated scoring**. For the **validated scoring**, it was 816 seconds with a coalition size of 16.

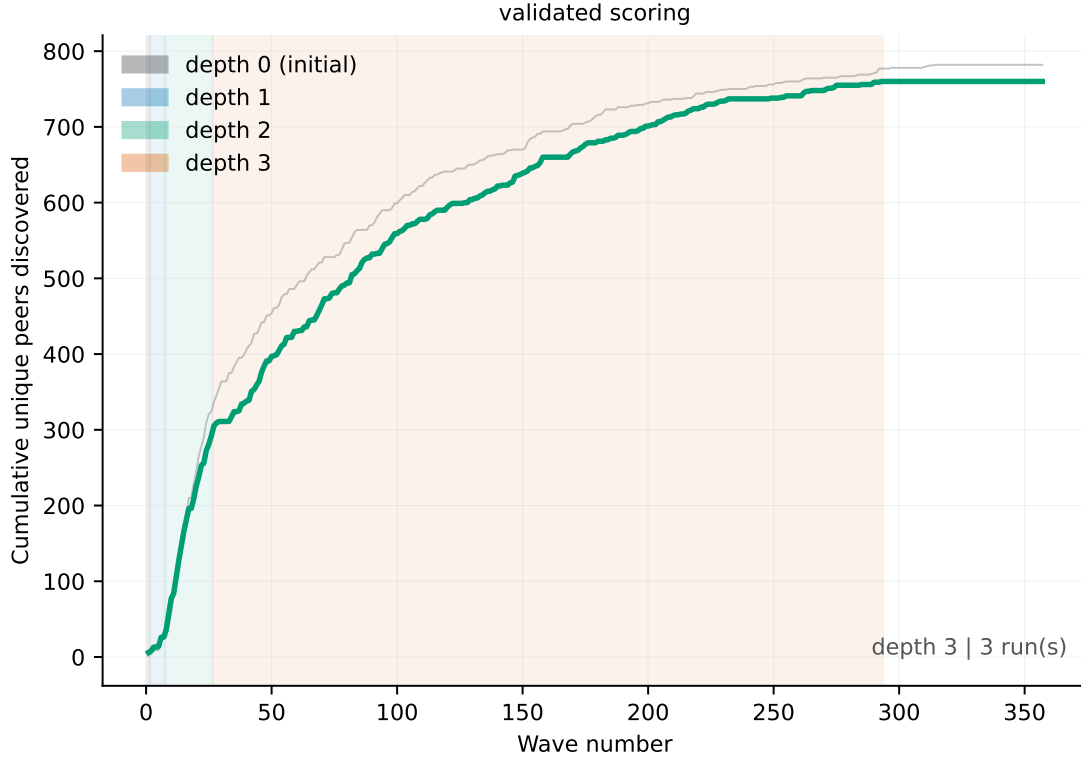


Figure 5.14: PX flood - limit exploration wave progression (median across seeds)

For **validated scoring**, Figure 5.14 shows that there is significant inter-seed variance. One of the three seeds stops at a 10% discovery rate after exhausting all found PX entries. The other two seeds reach 97 - 99%, however, in this case, depth 3 makes a significant difference. We can also see that the simulation time exceeds the potential of the inference, and that it exhausts all candidates in the PXs. Given the better protection offered by the degree parameters of the **validated scoring** scenario, the inference is more likely to fall into a bad path from the start and stop prematurely.

5.4.4 Topology drift

The PX flood inference method discovers 31% of the GossipSub topology using a coalition of 8 attackers with a depth of 1 in a single experiment under **baseline** scenario, at the cost of low precision per target due to the large size of the PX lists. The overall precision, however, is 100%, no nodes are invented by the inference. The attack is therefore satisfactory, but one point remains to be verified: to what extent the network is disrupted. In Figure 5.15, we can see that this disruption always remains below 6.5%. We need to put this figure into perspective: it is

6.5% on a global scale, since it is a decentralized network, no one has the means to measure this globally in a real network, so these few percentage points are virtually undetectable in reality. Nor does all the drift come from inference, since the comparison is made between the network before and after inference, part of the drift therefore also stems from the network’s natural evolution. The drift is not negligible, but it is not a prohibitive cost compared to the information gained. One can also argue that the information gained is not overly compromised by the inference process itself.

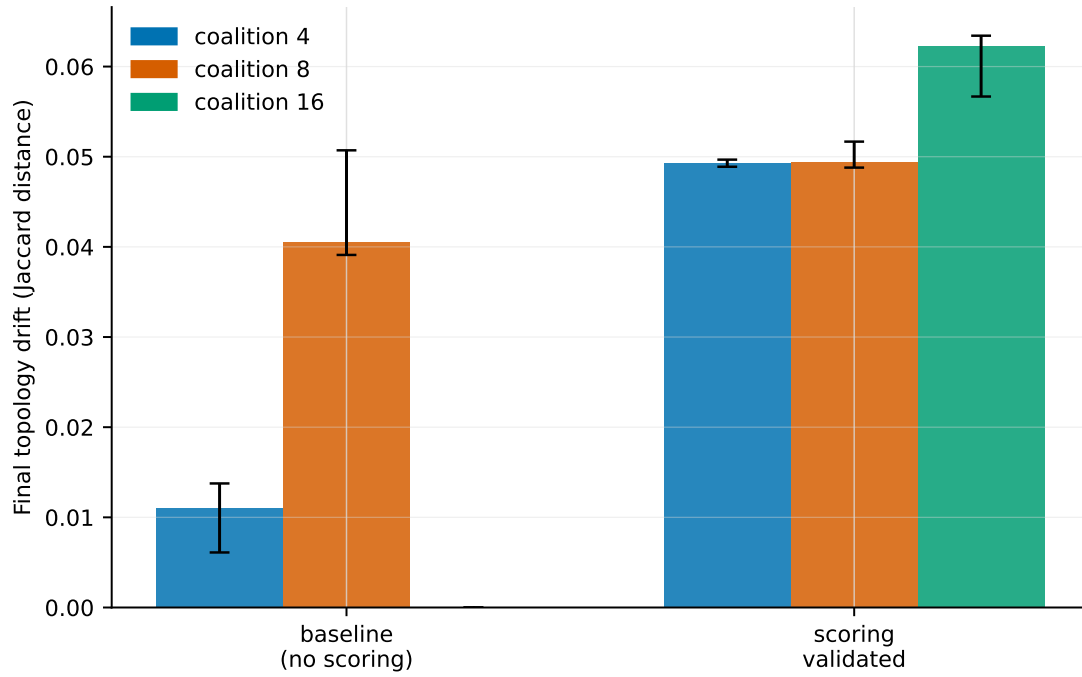


Figure 5.15: PX flood - topology drift depth 1 (median across seeds)

When the depth is set to 3 and nearly 99% of the network is discovered, the topology drift becomes very large: 45% for **validated scoring** with 16 nodes in the coalition. The drift is therefore proportional to the duration of the attack and thus to the amount of information obtained. Discovering the entire network thus involves significantly altering its structure.

5.4.5 Discovery bias

It is interesting to note that, within a limited amount of time, we can discover just under half of the network. But among these discoveries, do we discover nodes across the network uniformly, or do we primarily discover the most connected

nodes in the network? The initial intuition might be that high-degree nodes will be overrepresented in our discovery because they will appear more frequently in the PX lists of other nodes.

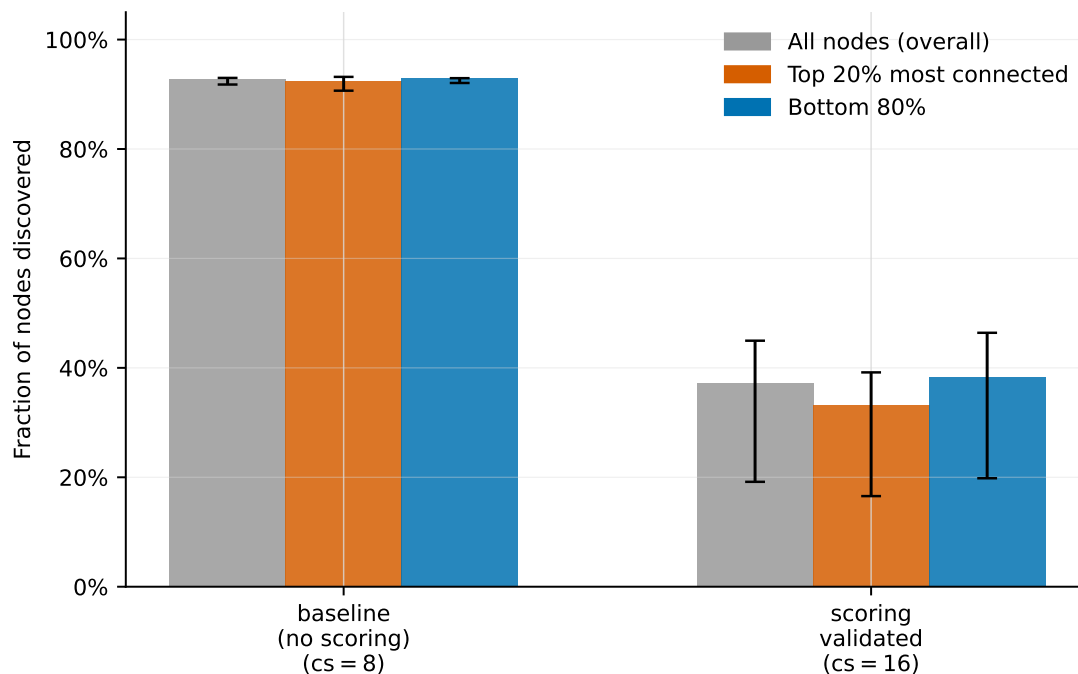


Figure 5.16: PX flood - degree bias (median across seeds)

Figure 5.16 shows the results for each scenario at depth 2. In the **validated scoring** scenario, as shown in Figure 5.10, we identify approximately 37.2% of the nodes in the final graph. However, we identify only 33.1% of the top 20% most connected nodes in the graph. We identify 38.2% of the remaining 80%. Therefore, we do not observe any bias in the most connected nodes of the network. For the **baseline**, the results are even more uniform, we see no bias here either, despite the high coverage of this inference. Thus, no bias appears even with increased coverage. It can be explained by how PX works. The PX list largely reveals the target's connected-reserve (minimum $D_{high} - D$), consisting of known peers not yet in the mesh. However, high-degree nodes are those already included in their neighbors' meshes. Note that this result may also depend on the various discovery methods used on top of GossipSub.

5.5 PX flood on the real GossipSub

After testing this attack on the simulator and successfully performing a large-scale inference, the ideal way to validate this attack is to test it on a smaller scale on Grid5000 using the actual GossipSub from libp2p. The inference was therefore tested on a network of 100 nodes with a coalition of 4 nodes. In this real network, D is set to 6 and D_high to 8. In theory, therefore, two nodes from the coalition are needed to fill the mesh. In this real network, there is an important abstraction: each node in the network has access to the global list of network nodes to facilitate initial connectivity. It is therefore not relevant here to measure the attack's precision or recall, as the attacked node can select any nodes it wishes to construct its PX list. The objective is rather to validate the feasibility of the attack against the actual protocol.

Since the network is smaller here, 100 nodes, we perform a simpler variant of the attack. Instead of traversing each node in the acquired PX lists in depth, the coalition will attack the same node multiple times in a row to exhaust its available nodes. When a node is PRUNED, there is a backoff period, a time during which the node cannot attempt to connect. The coalition must therefore wait out the backoff period and then try again.

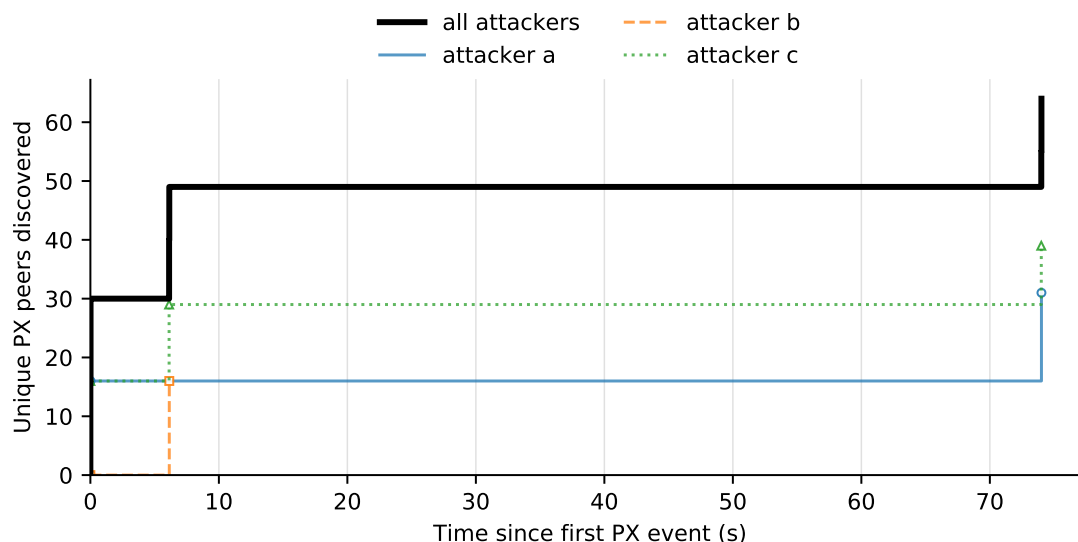


Figure 5.17: Real PX flood - nodes discovery over time

Figure 5.17 shows that three out of four nodes received one or more PXs from the target node. Node A received two PX lists, node B received one PX list, and node C received three PX lists by the end of the attack. This brings the total

number of PX lists for the coalition over time to six.

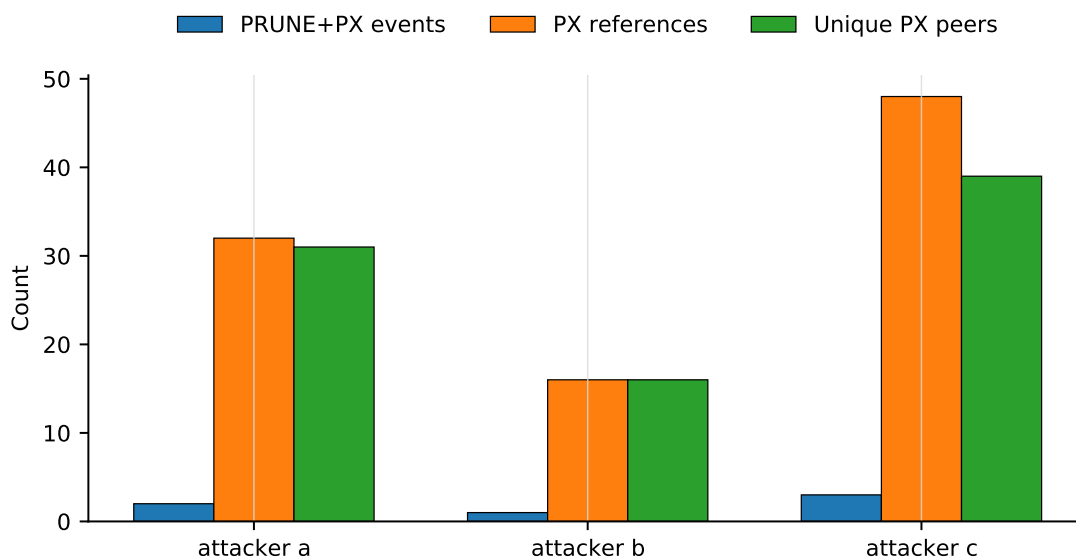


Figure 5.18: Real PX flood - statistics about inference

From these PX lists, shown in Figure 5.18, we can see that there were a total of 85 pairs. The orange bar represents the maximum number we could have received in each case, since some PX lists were incomplete. Of these 85 pairs, we obtained 64 unique pairs.

The inference is a success, even though the PX lists are not composed in a realistic manner, we can see that such an active inference on the actual protocol is possible and relatively simple to implement.

5.6 Countermeasures against PX flood

5.6.1 Defense design

Three defenses have been added to the simulator to try to mitigate this inference.

The first is a **PX list acquisition backoff**. This is an individual protection measure. After providing a PX list to a peer P for a topic T , the node no longer attaches a PX list to its PRUNE messages for that peer P for 200 heartbeats, which is more than three times the duration of the standard PRUNE backoff of 60 heartbeats. This threshold exceeds the normal backoff to prevent a node from repeating a PX-related inference on the same target as soon as it can rejoin the mesh. Thanks to this defense, the iterative nature of the inference is blocked, one cannot simply

exhaust a target's known peers by repeating GRAFTs as quickly as possible. This defense does not disrupt the initial operation of GossipSub because, for a legitimate node, a single PX list is sufficient for it to rebuild a mesh, and it does not need to immediately retry GRAFT on the same node again.

The second is an **overall throughput limit**. This mechanism caps the total number of PX peers that a node can provide to any requesting node over a sliding window of 30 heartbeats. The budget is D_high multiplied by 3, which is 24 peers for the default settings (D_high set to 8). When the budget is exhausted, any subsequent PRUNE within the same window is sent with an empty PX list, regardless of the requester. This defense primarily protects against large coalitions and ensures that the size of the coalition does not affect the maximum number of peers provided within a defined time frame. A PX list is already limited to 16 peers, but without this second limit, requesting multiple PX lists would allow this first limit to be bypassed. This defense also does not disrupt the normal operation of the protocol, as it allows for a normal flow of peers. A node might arrive right at the moment of an inference and thus during a shortage, but that same node can either redirect to another bootstrap node or try again a minute later to get a few peers instead of an empty list.

The third limit is a **flood detection** mechanism. It counts the number of incoming GRAFT messages per topic over a window of two consecutive heartbeats. When this counter reaches or exceeds $D_out + 2$, the node detects a potential coordinated flood and disables the transmission of PX lists containing data with a PRUNE for a configurable duration. Just like the two previous protections, the protection thresholds are configurable to best adapt to the traffic and real-world conditions of a potential application using GossipSub. If configured correctly, this final defense should not trigger false positives.

It is important to note that PX list is not intended to be a standalone system on which nodes rely for peer discovery. It is a quick way for a node to rebuild its mesh in the event of an unexpected PRUNE. If no PX is available, each node must be able to rely on an external discovery system to ensure its continuity.

5.6.2 Effect on global coverage

As shown in Figure 5.19, at a depth of 1, the defense is extremely effective, and the global coverage drops drastically. The coalition does not find a single node in PX because the **flood detection** defense is activated, thereby preventing any PX list from being attached to a PRUNE. At a depth of two, the result will not change significantly, for the baseline, it will remain at 0 since nothing was found initially. For **scoring validated**, it will expand slightly, but the scope and speed of inference are greatly limited. When comparing Figure 5.9 to coverage without defense, we can see the magnitude of the difference between the two cases.

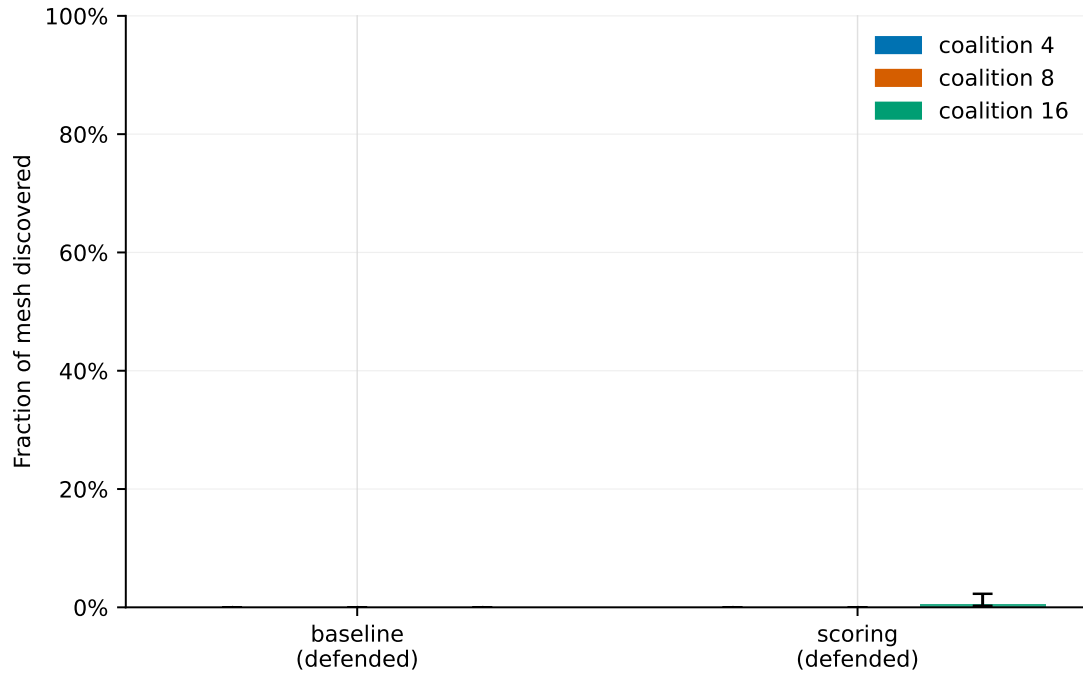


Figure 5.19: PX flood defended - coverage depth 1 (median across seeds)

5.6.3 Effect on attack progression

Figure 5.20 shows that, with defenses in place, the inference process stalls as early as the first waves, or, at best, never even begins. We can also compare this with the initial case in Figure 5.13 and see that of the three inferences that had yielded results, two were completely blocked, and one was severely limited.

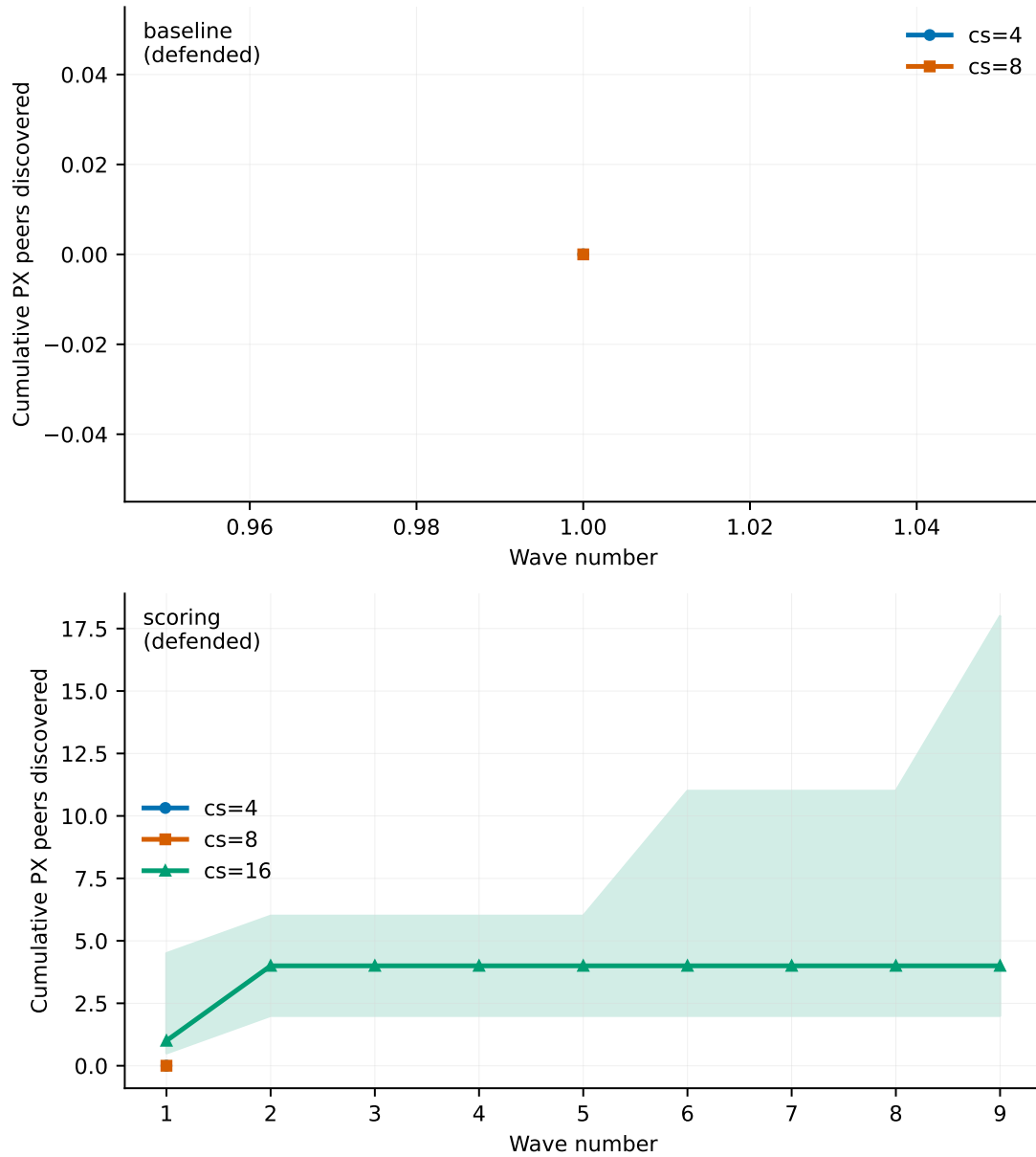


Figure 5.20: PX flood defended - wave progression (median across seeds)

5.6.4 Limits

Here, as shown in Figure 5.21, we can see that the defenses are not foolproof. At a depth of 3, one of the three seeds manages to start its inference and advance it far, while the other two are stopped by the defenses. To address this, it is possible to tighten the thresholds of the various defenses, however, this action can also lead to

false positives in detections and thus block legitimate traffic. For deployment in a real network, it would be useful to conduct tests to determine the ideal thresholds based on the objectives and requirements of the application using GossipSub.

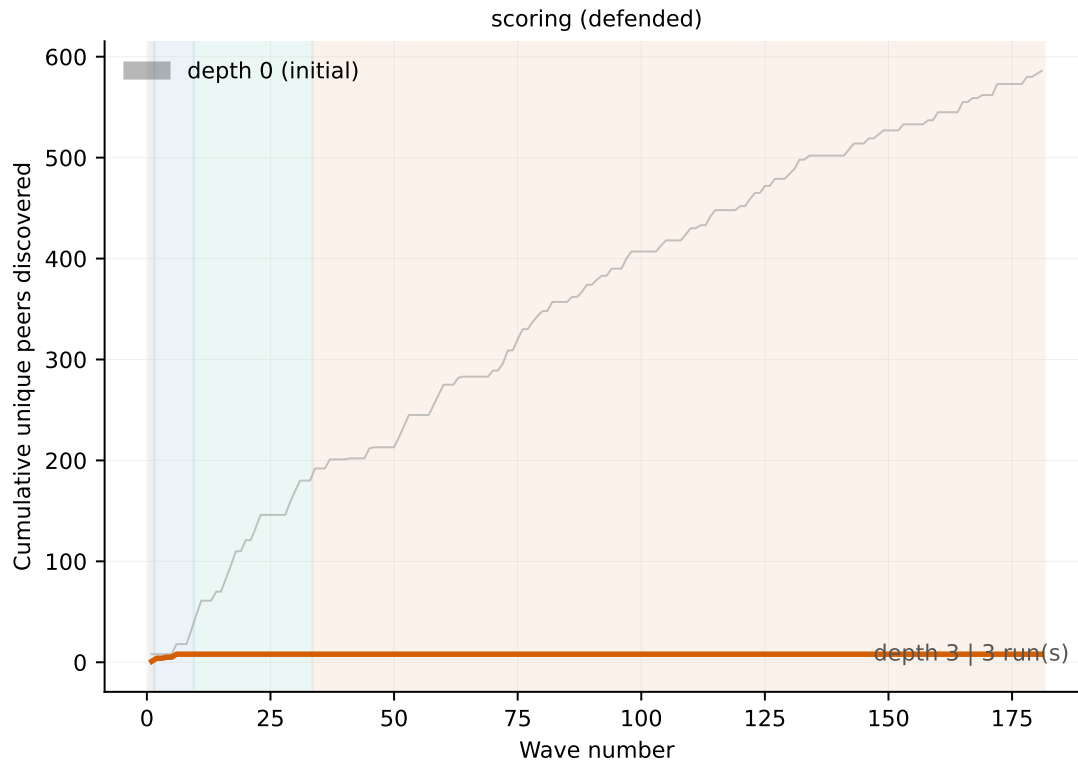


Figure 5.21: PX flood defended - limit exploration wave progression (median across seeds)

Regarding the limitations of the defenses, one might think that an attacker seeking to infer information simply needs to spread out their inference and **GRAFT** operations over more heartbeats to bypass the defenses. However, the defenses still achieve their objective: to slow down the process. Often, the goal is to be able to infer information quickly in order to leverage the topological information and then launch a targeted attack.

A current limitation of GossipSub is that as long as a node does not reach D_{high} , it does not tend to return to D via **PRUNE**, it only does so when it reaches the limit. If an attacker waits long enough between each **PRUNE**, they can thus avoid flood detection, but once again, the attack is greatly slowed down.

5.7 Discussions

The three inference methods tested in this chapter form a progression. Each one attempts to overcome the limitations of the previous one. Passive ego-mesh inference has shown that a node can accurately reconstruct its immediate surroundings. Passive coalition extended this capability to reconstruct a larger portion of the network and demonstrated that the combination of local views scales linearly. However, this inference cannot extend beyond the immediate neighborhood of the observers. PX flood breaks this limitation by actively acquiring topological information from nodes outside the coalition.

The results of the two passive inferences represent lower bounds on what is possible: in this case, they are very short-term experiments. In a real network, a node or a coalition of observer nodes left in place for several months would, through the network’s natural churn, observe many more nodes than those present in their mesh at any given time. They will accumulate the new nodes that join their mesh over time, maintaining an ever-growing list of nodes they know. This list can serve as a first step toward subsequently attempting PX-flood inference in a network. Once a large list has been obtained, active inference can be attempted on all the nodes found until a node with PX enabled is encountered.

When comparing passive and active inference, the striking difference lies in scope. Both passive methods achieve perfect precision and recall within their observable boundary: a coalition, regardless of size, reconstructs its boundary with $F1 = 1.0$ in a very short time. However, the limitation stems from the very structure of the method: a passive observer can only see the edges incident to its own nodes, and no signal in GossipSub reveals the edges between two nodes that are not part of the coalition. A coalition’s passive coverage therefore evolves at the rate of new neighboring nodes being added. This limitation makes global coverage unattainable without a coalition of an unrealistic size.

PX flood inference solves this problem. By triggering rejection PRUNEs on nodes, it obtains indirect topological information: a sample of the target’s extended neighborhood. Coverage of 31% with a coalition of 8 nodes is a significant improvement over the coverage of around 4% achieved with a passive coalition of the same size for **baseline** scenario.

Regardless of the method or configuration, an inferred node is a node that actually exists at the time of observation in the global topology, inferences never invent nodes, and there are no false positives. This is an important property: an attacker performing such an inference can be certain that the inferred nodes exist. In a real-world network, given that there is overall churn, if an inferred node leaves the network immediately after inference, accuracy may drop slightly. Where precision struggles more is with PX flood inference, which attempts to guess a node’s specific neighbors based on its PX list. Because the list is larger than

the size of a mesh, some elements are inevitably false positives, resulting in a significant limitation on precision. PX flood therefore allows us to identify nodes in the network, but not to reconstruct topological links with sufficient precision to be reliable.

Another observation, the analysis of degree bias, shows that the nodes discovered by the PX flood method do not exhibit a preference for high-degree nodes. This can be explained by the same mechanism: PX largely reveals pairs from the connected component that are nodes not overrepresented in the networks. Consequently, the inference is not likely to find strategic high-degree nodes.

Given our results, we should also consider the role of scoring in inference. One might expect that more robust scoring configurations would make inference more difficult by introducing more churn or disturbances. The results cast doubt on this hypothesis. As for passive inference, scoring makes no difference since the observers are legitimate participants in the network, therefore, there is no reason for scoring to penalize them. As for active inference, the difference does not really lie in scoring itself either, since the coalition's nodes do not join a mesh to misbehave, they fill the mesh so that there is no more room inside. That is why this inference is so interesting: it is not based on malicious behavior, but on legitimate behavior involving multiple nodes. The difference in security therefore lies rather in the gap between D and D_high , leaving more or fewer free slots in the mesh until it is filled. And thus allowing more or fewer coalition nodes to be surplus and to win the PX list with a rejection PRUNE. However, the results related to scoring must be viewed with some caution: while the simulator replicates GossipSub's structural parameters, in reality, in a real network, discovery often relies on a system external to GossipSub itself. Furthermore, the network has been in existence for a long time, is stable and fully populated, so each node knows a large number of peers and is connected to multiple nodes for gossip. The network in our simulation allows for less diversity once the network is up and running, and offers fewer opportunities for churn among the network's meshes.

The security of a decentralized network therefore depends not only on GossipSub and its proper configuration, but also heavily on the proper configuration of the external discovery system. If GossipSub is configured correctly but the external system allows too many nodes in the network to be discovered or enables inferences to be made, then the network's security can still be compromised.

To understand the inference enabled by PX, it is important to provide some context regarding this feature. First, it is a feature that can be enabled or disabled on each node in a GossipSub network. The primary purpose of PX is to rebuild the mesh after a PRUNE event. It is not recommended to enable this feature on all nodes. Best practice is to enable it on highly connected nodes, trusted nodes, or bootstrappers. This is important because it is a sensitive mechanism, if a malicious

node sends a PX, it could recommend connecting to Sybil nodes. Most of the time, applications that use GossipSub have their own discovery system. One example is Ethereum, which uses discv5 as its discovery system. PX is therefore not intended to be enabled globally. Here, PX was enabled globally so that we could test its limits in the event of a misconfigured network, this does not reflect a properly configured deployment. The scope of our study is limited to GossipSub, which is why we do not explore complementary discovery systems to make our inferences.

Having 8 to 16 nodes in a decentralized network is actually much more accessible than it seems. In a network like Ethereum, only validators are required to stake²: a node participating in the gossip network without a validator role needs only a few low-cost virtual servers. It can easily forward messages, maintain an acceptable GossipSub score, and participate fully in the mesh. The actual cost of a 16-node coalition therefore boils down to bandwidth and a few dozen euros per month for hosting. That said, this inference clearly illustrates the importance of degree settings on topological security. When the gap between D and D_{high} is greater, the attack requires more nodes to saturate a target’s mesh, which mechanically increases its cost. As is often the case in computer security, the real question is not whether a system is infallible, but what cost an attacker is willing to incur and whether the attack is worth that investment. In a peer-to-peer network, this logic is simple: the more nodes an attack requires to be effective, the less financially viable it is.

The precision and recall results for PX Flood should be considered optimistic estimates. Without global churn, every discovered node remains reachable throughout the experiment, which is not the case in a real network where some peers would have disconnected between waves, reducing the effective recall and introducing unreachable nodes into the inferred topology.

One result that remains robust despite all these caveats is the absence of degree bias: the fact that PX Flood does not preferentially discover the most connected nodes depends solely on the peer selection mechanism in the PX lists, and is not affected by churn or the realism of the scoring. Furthermore, the lack of complete scoring fidelity does not impact the very functioning of active inference, since the coalition does not need to misbehave, it just needs to fill a mesh, which is not considered misbehavior in GossipSub. Therefore, more accurate scoring would not change the efficiency of inference in the simulator.

The results obtained by the defenses show that they are effective against fast and aggressive inference, however, one in three seeds succeeds at depth three, which means that while the defenses are not foolproof, they do significantly slow down the process. An attacker whose goal is to prepare a targeted attack (eclipse [7], deanonymization [9]) needs to act quickly, and the defenses specifically reduce

²To be validator in the network, a node must stake a certain amount of cryptocurrency.

this time advantage. It is possible to strengthen the defenses by tightening the thresholds, but this is a double-edged sword because it risks blocking legitimate traffic. It is always important to balance the risks that are acceptable to take and configure the defenses accordingly. The best defense will always be to disable PX lists globally if they are not useful or essential to the network.

Chapter 6

Related Work

6.1 Early P2P networks and topology measurement

In the early 2000s, consumer-oriented P2P networks emerged, led by Gnutella [5]. These large-scale networks are characterized by their decentralized and dynamic nature. These systems became popular because they relied on no central authority, and there was no global view of the network. These networks do not operate directly on a physical network, but instead build a logical overlay upon which they rely for the transmission of information.

Given the absence of a global view, the topology is opaque. Studies very quickly turned to the discovery of the topology of these networks. The central reason for this research is performance: if we know the topology, we can better understand the factors that influence network performance: information propagation, latency or network load. However, the topology is not directly observable: it must be inferred from local observations. Each node has a local view, we must make use of this partial view.

To do this, several techniques are used, notably active crawling. This technique involves joining the network as a node and gradually discovering its neighbors in order to reconstruct a connection graph. And this is done recursively: once we have neighbors, we try to determine the neighbors of those neighbors, and so on. From several local observations, we can gradually arrive at a global approximation.

In 2002, Ripeanu et al. [5] studied Gnutella by attempting to carry out large-scale mapping to observe various properties: node degree distribution, heterogeneity, etc. They built an active crawler that enabled them to reconstruct the network. The crawler joins the network as a peer and uses PING/PONG messages to discover neighbors. Thanks to this mechanism, the crawler gradually reconstructs the network. However, they encountered some difficulties during crawling: the network

is dynamic, it is not static over time, so it is harder to obtain an accurate picture of the network because it is constantly changing. Despite this, they succeeded in producing results showing that the topology can be reliably approximated. They highlight a critical point: the overlay used by Gnutella is far removed from the physical reality of the network, leading to performance issues. There is a huge amount of unnecessary inter-AS traffic that could be avoided, resulting in latency. This paper demonstrates that the topology of a large P2P network can be inferred using an active crawler, despite the network’s dynamic and decentralized nature.

Later, in 2007, Hamra et al. [6] worked on BitTorrent and also attempted to reconstruct the network by gathering peers via direct connections. They managed to partially reconstruct the graph. Unlike Gnutella, the structure of the BitTorrent network is influenced by coordination mechanisms. There is greater control over the formation of the topology. In this paper, Al Hamra et al. [6] also examined the impact of the peer exchange that occurs in BitTorrent on the overlay. They showed that this results in a chain-like structure with a large diameter. As early as 2007, PX was not a trivial mechanism and was already influencing topology observability.

At that point, topology was already at the centre of the debate in P2P networks because it has an impact on resilience, scalability and, above all, performance. A poorly formed network can cause slow propagation of information or unnecessary network overload. Understanding the topology means having the ability to improve network performance.

However, knowledge of the topology does not merely enable better performance, it can also be exploited for malicious purposes to target weak points in the network. Controlling the topology means controlling the information.

6.2 Modern large-scale P2P systems

Since previous studies, P2P networks have evolved but retain the same fundamental properties: they are dynamic, logical overlay networks, decentralized and based on local views. The protocols underpinning them are no longer the same, but they retain the same fundamental properties as older P2P networks such as Gnutella or BitTorrent.

One of the first characteristics to change is the difference in scale, these networks are now much larger and more heterogeneous. Participation is open to anyone who wishes to join: this is a double-edged sword. The networks are growing, but they open the door to malicious participants. There is greater diversity in behavior. The lack of an overall view persists, and is exacerbated as the networks grow in size. The increase in size also creates a larger attack surface.

Crawling still exists despite the evolution of P2P networks, in 2020, Henningsen et al. [25], [26] studied crawling on the IPFS network. Using a crawler that relies

on discovering peers via IPFS protocols, they carried out a partial reconstruction of the network. Given the network's size, it is very difficult to reconstruct it in its entirety. But size is not the only factor: the network's dynamism also comes into play and complicates crawling. Based on this observation, they were able to assess the network's health: connectivity, structure and distribution. Topology is therefore always a key consideration, it must be analyzed to evaluate a peer-to-peer network. However, its analysis has evolved due to the changes in scale that P2P networks have undergone since their inception.

Focusing on Ethereum [2], it is a blockchain that relies on a P2P network for propagation. Like any blockchain, its principle lies in the dissemination of transactions and blocks. Rapid and efficient propagation is required: this depends on the network and therefore on the topology. Better connections mean better coverage and lower latency. Network performance determines system performance. If propagation is too slow, there is a risk of a chain fork (the blockchain splitting into two distinct possible states). Rapid propagation ensures the security of the consensus.

However, such a network is complex and extremely large, comprising many thousands of nodes in the case of Ethereum. This sheer scale makes it difficult to control and monitor. For such a network to function, a balance must be struck between efficiency and robustness.

To achieve this balance, efficient protocols are required: this is where GossipSub comes into play. GossipSub is designed to address these constraints through its hybrid mesh + gossip architecture, which reduces bandwidth usage and improves robustness. Inference therefore remains relevant today and is all the more so with the increasing size of networks and new protocols such as GossipSub.

6.3 Privacy and deanonymization

A P2P network does not necessarily mean anonymity. Whilst no one is supposed to have a view of the entire network, the network level does reveal identity information about the nodes. Connecting to a node requires knowing its IP address, which is sensitive information. A participant in the network cannot remain completely anonymous: this information is known to its immediate neighbors in the overlay.

One attack not mentioned previously that targets the topology is the de-anonymisation attack. The aim is to reconstruct part of the topology and link each node's network identity (an IP address, for example) to specific entities in order to understand who is behind the network. This reconstruction can be carried out actively (via crawling) or passively by observing the communication. By analyzing information propagation patterns, hypotheses can be formed about the position of nodes in the network, as well as their identity.

As demonstrated by Heimbach et al. [9], it is possible to deanonymize Ethereum validators using only network observations. Their approach is based on the correlation between the behavior of nodes, their network activity and their position within the topology.

In Ethereum, validators are partially identifiable, which makes it possible to draw correlations between network activity and their role within the system. Thanks to this, it is possible to locate certain nodes or infer information.

De-anonymization is not limited to Ethereum and is not a new problem in distributed systems. As demonstrated by the work of Fanti et al. [27], de-anonymization can also occur on the Bitcoin network, which offers only a low level of anonymity during participation.

The topology therefore takes on a new dimension, it is no longer merely a means of understanding the network's performance or robustness. It contains sensitive information. A node's position within the network, its communication methods and its connections can be used to reveal information about its role or identity. Inference of the topology thus becomes a potential vector for information leakage.

Chapter 7

Conclusions

This work analyzed the extent to which the topology of a GossipSub network can be inferred through passive observation or active participation. Three methods were implemented and evaluated in a validated GossipSub simulator, across two scenarios and multiple initializations.

The first result is that a passive observer can reconstruct its local mesh in approximately three heartbeats after joining the network. **GRAFT** and **PRUNE** messages provide evidence of the mesh edges, and no configuration prevents this local reconstruction. We thus have a baseline: every node participating in a GossipSub network has full access to its immediate neighborhood.

Next, we observe that the combination of passive observers is linear. A coalition of k nodes adds approximately D new visible nodes per observer, achieving an additional coverage of roughly 4% to 6% for $k = 8$. Passive inference is limited by the coalition’s boundary because it lacks additional mechanisms to infer edges beyond its boundaries. The advantage of the passive coalition lies in its invisibility: it generates no unusual traffic and leaves no trace.

This leads us to the fact that **PX** flood extends the reach that the passive coalition lacked. A coalition of 8 discovers 31% of the overall topology in **baseline** scenario. The overall accuracy of 100% makes the inference interesting. However, the accuracy of edge reconstruction between inferred nodes is very low. We can infer which nodes are part of the network, but we infer the relationships between nodes imprecisely. We also see the importance of properly configuring degree parameters, which in this case allows us to significantly limit the inference.

More specifically, it is the difference between D and D_{high} that is the determining factor: the smaller this difference, the more nodes an attacker must have to saturate a target’s mesh and trigger a **PRUNE** accompanied by a **PX** list. Scoring itself is not the main obstacle to active inference: it is the configuration of the degree parameters that truly controls its effectiveness.

Furthermore, an analysis of discovery bias reveals a counterintuitive result: the

PX flood does not favor the most connected nodes in the network. PX lists primarily reveal nodes from a target’s connected pool, which are precisely the nodes that are less represented in their neighbors’ meshes. An attacker seeking to target strategic nodes cannot therefore rely on this method to identify them preferentially.

This final inference highlights an important metric: topology drift during active inference. It is clear that the more waves there are in the inference process, the greater the drift becomes, therefore, a balance must be struck between achieving the desired coverage without overfitting, so as not to disrupt the network.

These results were also compared against a real-world implementation of libp2p/GossipSub on Grid5000. The attack was conducted on a network of 100 nodes and proved successful, confirming that this is not behavior specific to the simulator, but rather a property of the protocol itself.

To limit the effectiveness of this inference, three countermeasures were designed and tested: a backoff on PX list acquisition per peer, a limit on the total number of peers provided within a time window, and a flood detection mechanism. At depths 1 and 2, these defenses almost completely block the attack. At depth 3, one in three inferences still manages to progress before being stopped. These defenses are not intended to make inference impossible, which would be difficult to guarantee, but to slow it down sufficiently so that the information obtained becomes outdated before it can be used.

Taken together, these results answer the central question: Is the topology of a GossipSub network partially reconstructable, and if so, with what degree of precision? Passive inference allows an attacker to perfectly identify their neighbors at negligible cost. Active inference extends this knowledge to a significant portion of the overall topology. This knowledge is sufficient to identify key nodes, potentially de-anonymize them, and prepare local attacks. However, the inferred topology is probabilistic and contains many incorrect links. Furthermore, this active attack relies on a poorly configured network. In the case of a correct configuration, this inference can still be performed but on a smaller scale.

Finally, it is helpful to put this threat into its practical context. In a network like Ethereum, setting up a coalition of 16 nodes requires only a few basic virtual servers, at a cost of a few dozen euros per month.

These results do not imply that GossipSub is vulnerable, but rather that the protocol’s signals constitute a channel of information about the topology, and that if this is combined with a misconfiguration or a specific application usage, it can lead to meaningful and potentially dangerous inferences regarding network security.

7.1 Future work

This work can be extended in several interesting directions.

First, the tests can be conducted on networks of realistic size. All experiments were performed on networks of 1,000 nodes. The main Ethereum network, for example, is much broader than that. It would be interesting to scale the inference campaigns to this order of magnitude to verify how coverage rates change or whether they hold true at a larger scale. We should observe whether the redundancy that limits coverage after a certain point becomes more pronounced or less prevalent as the network grows.

Next, it would be interesting to explore multi-topic inference. By design, GossipSub is intended to be used on multiple topics simultaneously. In this thesis, we studied reconstruction for a single topic. However, since a node can participate in multiple topics, performing inference across multiple topics simultaneously would allow for the cross-referencing of information and improved node coverage. Especially since a real-world Ethereum DAS [13] (Data Availability Sampling¹) deployment operates with hundreds of subtopics, the potential for cross-referencing information is all the greater.

An interesting extension would be not to perform an inference at a given moment, but to try to keep the topological information up to date by studying how inference results degrade as the mesh evolves. And to use this to determine the frequency of re-inference required to maintain sufficient quality.

On the defense side, several approaches can be explored, such as implementing the proposed defenses from the simulator in the actual GossipSub. An additional defense could be to penalize a node's score in the event of malicious behavior related to PX flooding. This addition could enable a genuine long-term topological impact resulting from such attacks. The impact on the score would need to last a long time so that the affected nodes are gradually excluded from the network.

Next, inferences based on other signals could be tested. This thesis focuses primarily on **GRAFT**, **PRUNE**, and **PX**. However, we could also use **IHAVE**, **IWANT**, and network messages. For example, we could send a message with specific content and wait to receive it back in order to make temporal correlations and infer the distance between two nodes. We can also make inferences: if we observe that a node in our vicinity shares the same **IHAVE** messages that we received from a known peer, we can infer that this peer is also part of its mesh. There are still other possible passive inferences based on these signals.

Finally, the simulator could be improved to incorporate the actual scoring system and potentially make it asynchronous, thereby allowing for better testing of inferences without the bias introduced by a synchronous network.

¹Data Availability Sampling is a technique that allows nodes to verify the publication of data in a block without having to download the entire block, by randomly sampling small fragments. It is central to Ethereum's scaling road-map.

Bibliography

- [1] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008.
- [2] G. Wood *et al.*, “Ethereum: A secure decentralised generalised transaction ledger,” *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.
- [3] D. Vyzovitis, Y. Napora, D. McCormick, D. Dias, and Y. Psaras, “Gossipsub: Attack-resilient message propagation in the filecoin and eth2. 0 networks,” *arXiv preprint arXiv:2007.02754*, 2020.
- [4] J. Benet, “Ipfs-content addressed, versioned, p2p file system,” *arXiv preprint arXiv:1407.3561*, 2014.
- [5] M. Ripeanu, I. Foster, and A. Iamnitchi, “Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design,” *arXiv preprint cs/0209028*, 2002.
- [6] A. A. Hamra, A. Legout, and C. Barakat, “Understanding the properties of the bittorrent overlay,” *arXiv preprint arXiv:0707.1820*, 2007.
- [7] A. Singh, T.-W. Ngan, P. Druschel, D. S. Wallach, *et al.*, “Eclipse attacks on overlay networks: Threats and defenses,” 2006.
- [8] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, “Eclipse attacks on {bitcoin’s}{peer-to-peer} network,” in *24th USENIX security symposium (USENIX security 15)*, 2015, pp. 129–144.
- [9] L. Heimbach, Y. Vonlanthen, J. Villacis, L. Kiffer, and R. Wattenhofer, “Deanonymizing ethereum validators: The {p2p} network has a privacy issue,” in *34th USENIX Security Symposium (USENIX Security 25)*, 2025, pp. 1319–1338.
- [10] libp2p, *Go-libp2p: The go implementation of the libp2p networking stack*, version 0.48.0. [Online]. Available: <https://github.com/libp2p/go-libp2p>.
- [11] DataHop Labs, *Kademlia-simulator: A kademlia simulator based on peersim, peerdas branch*. Repository our simulator is forked from. [Online]. Available: <https://github.com/datahop/kademlia-simulator/tree/peerdas>.

- [12] A. Montresor and M. Jelasity, “Peersim: A scalable p2p simulator,” in *2009 IEEE Ninth International Conference on Peer-to-Peer Computing*, IEEE, 2009, pp. 99–100.
- [13] M. Pigaglio *et al.*, “Pandas: Peer-to-peer, adaptive networking for data availability sampling within ethereum consensus timebounds,” *arXiv preprint arXiv:2507.00824*, 2025.
- [14] CloudLargeScale-UCLouvain, *Pandas: Simulator and prototype experiments*, Artifact repository of the PANDAS paper. Go/libp2p prototype forked for our GossipSub experiments on Grid5000. [Online]. Available: <https://github.com/CloudLargeScale-UCLouvain/PANDAS>.
- [15] A. Kumar, M. von Hippel, P. Manolios, and C. Nita-Rotaru, “Formal model-driven analysis of resilience of gossipsub to attacks from misbehaving peers,” in *2024 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2024, pp. 2142–2160.
- [16] U. Pavloff, Y. Amoussou-Guenou, and S. Tucci-Piergiovanni, “Byzantine attacks exploiting penalties in ethereum pos,” in *2024 54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, IEEE, 2024, pp. 53–65.
- [17] libp2p. “Gossipsub v1.0: An extensible baseline pubsub protocol (libp2p specifications).” [Online]. Available: <https://github.com/libp2p/specs/blob/master/pubsub/gossipsub/gossipsub-v1.0.md>.
- [18] A. Demers *et al.*, “Epidemic algorithms for replicated database maintenance,” in *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, 1987, pp. 1–12.
- [19] A.-M. Kermarrec and M. Van Steen, “Gossiping in distributed systems,” *ACM SIGOPS operating systems review*, vol. 41, no. 5, pp. 2–7, 2007.
- [20] Y. Psaras and D. Dias. “Gossipsub: An attack-resilient messaging-layer protocol for public blockchains,” Protocol Labs Research. [Online]. Available: <https://research.protocol.ai/blog/2020/gossipsub-an-attack-resilient-messaging-layer-protocol-for-public-blockchains/>.
- [21] libp2p. “Gossipsub v1.1: Security extensions to improve on attack resilience and bootstrapping.” [Online]. Available: <https://github.com/libp2p/specs/blob/master/pubsub/gossipsub/gossipsub-v1.1.md>.
- [22] P. Maymounkov and D. Mazieres, “Kademlia: A peer-to-peer information system based on the xor metric,” in *International workshop on peer-to-peer systems*, Springer, 2002, pp. 53–65.

- [23] Ethereum. “Node discovery protocol v5 (discv5) specification. ”[Online]. Available: <https://github.com/ethereum/devp2p/blob/master/discv5/discv5.md>.
- [24] D. Balouek *et al.*, “Adding virtualization capabilities to the grid’5000 testbed,” in *International Conference on Cloud Computing and Services Science*, Springer, 2012, pp. 3–20.
- [25] S. Henningsen, S. Rust, M. Florian, and B. Scheuermann, “Crawling the ipfs network,” in *2020 IFIP Networking Conference (Networking)*, IEEE, 2020, pp. 679–680.
- [26] S. Henningsen, M. Florian, S. Rust, and B. Scheuermann, “Mapping the interplanetary filesystem,” in *2020 IFIP Networking Conference (Networking)*, IEEE, 2020, pp. 289–297.
- [27] G. Fanti and P. Viswanath, “Deanonymization in the bitcoin p2p network,” *Advances in neural information processing systems*, vol. 30, 2017.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | www.uclouvain.be/epl