Fault-Tolerant Information
Spreading Algorithms

Tariq Toukan
Fault-Tolerant Information Spreading Algorithms

Research Thesis

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Tariq Toukan

Submitted to the Senate
of the Technion — Israel Institute of Technology
Adar 5775 Haifa March 2015
This research was carried out under the supervision of Prof. Keren Censor-Hillel, in the Faculty of Computer Science.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Prof. Keren Censor-Hillel, for her devoted support, guidance and concern throughout this research. Thanks to my dear family for their love and support at all times. Thanks to Mohsen Ghaffari and Fabian Kuhn for useful discussions. Thanks to Shmuel Zaks and Yuval Emek for useful comments and feedback.

The generous financial help of the Technion is gratefully acknowledged.
# Contents

## List of Figures

Abstract 1

1 Introduction 3
   1.1 Distributed Computing ............................... 3
   1.2 Problem Description ................................. 4
   1.3 The Vertex-Congest Model ............................. 5
   1.4 Background and Motivation ............................. 5
   1.5 Related Work ........................................ 6
   1.6 Our Contribution ..................................... 8
   1.7 Thesis Roadmap ...................................... 9

2 Preliminaries 11
   2.1 Mathematical Background ............................. 12
   2.2 Time Complexities .................................. 14

3 The Uniform Random Algorithm 17

4 A Fast Information Spreading Algorithm 21
   4.1 Algorithm Description ............................... 21
   4.2 Time Analysis for $G_{n,k}$ .......................... 24
      4.2.1 Analysis of the Random Phase ....................... 26
      4.2.2 Analysis of Ranking Phases ....................... 28

5 Fault Tolerance 33
   5.1 Shuffle Phases .................................... 33
   5.2 Resilience to Faults ............................... 38

6 Discussion 41
   6.1 Summary ........................................... 41
   6.2 Static-Routes Algorithms ............................ 41
   6.3 Steady Growth Graphs ............................... 42

Hebrew Abstract i
# List of Figures

1.1 The nodes in red form a connected dominating set (CDS) ........................................ 6

2.1 Graph $G_{n,k}$ ........................................................................................................ 12

2.2 A shortest path of length $i+1$ between nodes $u$ and $v$ ........................................ 15

4.1 Phases of the algorithm .............................................................................................. 22

4.2 The Ranking Function ............................................................................................... 24

4.3 Illustration for expected routes of a message from $C_1$ to $C_{n/k}$ in Algorithm 4.1 ................................................................. 26

5.1 Phases of the modified algorithm ............................................................................... 34

5.2 Illustration for expected routes of a message from $C_1$ to $C_{n/k}$ in Algorithm 5.1 ................................................................. 34
Abstract

We consider a distributed system of $n$ nodes communicating in the synchronous Vertex-Congest model. In this model, in each round, each node sends the same message of size $O(\log n)$ bits to all neighbors. We investigate the problem of information spreading, where each node has an initial message, and the goal is to collect messages of all other nodes.

Focusing on communication graphs that are $k$-vertex-connected, we argue that since the existing near-optimal algorithm that requires $O(n \log(n)/k)$ rounds after some preprocessing uses static paths for routing, it becomes highly sensitive to failures. While combining paths together and using redundant routing makes the algorithm more resistant, the construction of the paths in the presence of failures is still an open problem. On the other hand, we show that the naturally-robust fully randomized algorithm is slow on a simple family of $k$-vertex-connected graphs, denoted by $G_{n,k}$, consisting of $n/k$ cliques of size $k$ that are connected by a path of matchings, requiring $\Omega(n/\sqrt{k})$ rounds.

We propose an algorithm that uses non-uniform randomization, with probabilities that change over time according to the execution. We prove that for $G_{n,k}$, our algorithm completes in $O(n \log^3(n)/k)$ rounds with high probability, and is resilient to independent failures that occur with large probabilities.
Chapter 1

Introduction

1.1 Distributed Computing

We investigate the problem of spreading information between nodes of a distributed system, consisting of distant components that communicate via passing messages in order to coordinate their actions. Each node in the network can communicate with its neighbors in the communication graph, and perform local operations. In a distributed system, there is no central entity that is aware of the states of all components and controls the execution accordingly, as opposed to the operation manner of a centralized system. Instead, the components communicate by sending messages according to a predefined protocol. To perform their tasks, the execution in each node proceeds by coordinating local actions with messages exchanged. Such systems are widespread due to their high processing abilities that match the needs of global internet services.

A distributed algorithm is one that is designed to run on a distributed system, typically with a large number of nodes. The main quality measures for such algorithms consider aspects of time complexity, communication complexity, fault tolerance and scalability.

Failure in a computer device is normally a rare event, and the expected time interval between two consecutive failures can be relatively long. However, such events become more frequent as the number of components in the system gets larger. In a typical distributed system, the number of nodes could be very large, which makes failure events more common and non negligible.

Many things might go wrong in a distributed setting: components and communication links might crash, malfunction or function maliciously, transiently or permanently. A fault-tolerant computer system is one designed to continue working to a level of satisfaction in the presence of faults.

In this thesis, we are interested in the problem of information spreading in a certain communication model of distributed systems, defined later, and focus on measures of time complexity and fault-tolerance.
1.2 Problem Description

Gathering data from nodes in a network is a central issue in distributed computing, as it is essential for many distributed applications that require the knowledge of some global data. It is the focus of many recent studies, driven by the growing need for distributed applications, such as file sharing [MM02], media streaming [DBGM01,PWCS02,ZLZY05,LMSW07], maintenance of replicated databases [DGH+87,FPRU90], publish/subscribe systems [EGH+03], group membership [KMG03,GK10], failure detection [VRMH98], resource discovery [HBLL99], data aggregation [BGPS06], modeling the spread of computer viruses [BBCS05], and more.

We investigate the problem of information spreading, where each node has an initial message, and the goal is to collect messages of all other nodes. The problem has been studied under different names such as gossip, rumor spreading and information dissemination. The difficulty level of the problem highly depends on the communication model and the network topology, and varies from one setting to another. For example, given the LOCAL model, the problem becomes trivial. In this model, in each round, every node can send messages of unbounded size to each of its neighbors. The simple algorithm in which every node repeatedly sends all the messages it knows spreads information in $D$ rounds, where $D$ is the diameter of the graph. This means that for $n$-node graphs, the runtime of the same algorithm might vary from only 1 round for a clique topology, up to $n - 1$ rounds for a path graph.

Interesting studies refer to more restrictive communication models, making the problem more challenging. In the widely studied GOSSIP model, nodes may only initiate contact with a single neighbor in each round and exchange messages of unbounded size. However, the lack of global information makes it difficult to make any sort of principled choice that is based on knowledge. Researchers show that the uniform gossip algorithm, in which each node selects a neighbor uniformly at random each round, requires $O\left(\frac{\log n}{\phi}\right)$ rounds$^1$, where $\phi$ is the conductance of the graph [Gia11]. Roughly speaking, the conductance of a connected graph is a value $\phi \in (0, 1]$ that captures the connectivity by being large for graphs that are well connected, and small for graphs that are not. The above bound means that the algorithm is fast on well-connected graphs, but slows down dramatically when it comes to dealing with bottlenecks. In [CHHKM12], Censor-Hillel et al. propose an algorithm that requires at most $O(D + \text{polylog}(n))$ rounds on all graphs, with no dependence on their conductance, being optimal up to at most an additive polylogarithmic factor. Notice that the lower bound of $\Omega(D)$ holds even for the less restrictive LOCAL model.

Other models, that were introduced to match actual applications, restrict messages to be of bounded size. The CONGEST model [Pel00, Chapter 2] is one that lets each node send a different message of size $O(\log n)$ bits to each of its neighbors, in each round. One of the algorithms proposed in [CHGK14a] runs in this setting. It constructs

---

$^1$In this thesis we consider the binary logarithm (base 2), unless stated otherwise.
fractionally edge-disjoint weighted spanning trees, and uses them to statically route messages over the graph. Each tree has a weight, and the total weight of trees in which an edge participates is bounded. Another algorithm proposed in the same paper disseminates information in the Vertex-Congest model, which is the model in the focus of this research.

1.3 The Vertex-Congest Model

In this thesis we focus on the Vertex-Congest model. In this model, in each round every node sends the same message to all of its neighbors. Each message is $O(\log n)$-bits long, a standard size that allows nodes to send an identifier (ID) in one message. As the name suggests, the congestion in this model is in the vertices, while in CONGEST it is in edges. To distinguish between the two models, we refer to the CONGEST model as the Edge-Congest model. The Vertex-Congest model is a restricted version of the Edge-Congest model, and thus any algorithm that works in the Vertex-Congest model works in the Edge-Congest model as well.

This model is applicable to wireless networks with reliable broadcasts. In general, reliable broadcast is not a trivial attribute when it comes to wireless communication, as multiple nodes might transmit packets simultaneously, causing collisions. A receiver that gets corrupted packets cannot recover any of the original content, and the packets should be retransmitted separately at a later time. However, reliable broadcasts can still be assumed when operating on top of an abstract medium access control (MAC) layer [KLN11], as it takes care of all these collision and retransmission issues.

We note that significant results for connected components identification [Thu95] and minimum-weight spanning tree [KP95], that were studied in the Edge-Congest model, work in the Vertex-Congest model. In addition, there are results in the Broadcast Clique model (e.g. [DKO14], [HP14], [CHP14]) in which the network is fully connected (forming a clique), which masks away the effect of distances on the computation and focuses on the limited bandwidth.

1.4 Background and Motivation

When looking for immediate time complexity bounds, we notice that according to our model there is no point in re-sending messages, as all links are reliable. In Lemma 2.9 we show an upper bound of $O(n)$ rounds for all reasonable algorithms, in which each node broadcasts each message at most once and does not remain quiet if it has unsent messages. On the other hand, we observe that the minimum vertex-cut of a graph is a bottleneck for the flow of information, and thus it implies a lower bound for time complexity, and an upper bound for the possible throughput. In this feasible time-complexity interval, it is reasonable to expect and aspire to higher throughput (and accordingly, lower time complexity) for well-connected graphs.
To give a measure for how well a graph is connected, we consider the parameter $k$ of the vertex-connectivity. An $n$-node communication graph is said to be $k$-vertex-connected if the graph resulting from deleting any (perhaps empty) set of fewer than $k$ vertices remains connected. Menger’s Theorem [Men27] gives an equivalent definition: a graph is $k$-vertex-connected if, for every pair of its vertices, it is possible to find $k$ vertex-disjoint paths connecting these vertices. A $k$-vertex-connected graph has minimal vertex-cut of size $k$, implying a lower bound of $\Omega(n/k)$ rounds for information spreading. A straightforward conclusion from Menger’s Theorem is that, in a $k$-vertex-connected graph it is possible to transmit $k$ different messages in parallel between a specific pair of nodes by utilizing these vertex-disjoint paths. However, this approach cannot be directly generalized to contain more vertices, as the $k$ disjoint-paths of some pair of nodes intersect with other paths of other pairs of nodes. A more suitable structure that covers all nodes and is capable of transmitting messages between them, from any source to any destination, is a connected dominating set (CDS) described next.

A dominating set of an undirected graph $G = (V,E)$ is a set of nodes $S \subseteq V$ such that every node $v \in V$ is either in $S$ or has a neighbor $u \in S$. A connected dominating set (CDS) is a dominating set that is connected (see example in Figure 1.1). An algorithm introduced in [CHGK14b, CHGK14a] constructs (fractionally) vertex-disjoint CDSs in a preprocessing stage and then utilizes them to broadcast messages in parallel. In our algorithm, without actually constructing CDSs, we get a behavior that is in a way similar to having $k$ CDS structures. They are dynamically developed throughout the execution bypass faulty nodes and work in combined groups of $O(\log n)$ to get high robustness, at the price of a poly-logarithmic factor in runtime.

1.5 Related Work

One approach for disseminating information that was introduced in [ACLY00] and has been intensively studied (e.g. [LYC03, HKM+03, DMC06, MAS06]) is network coding. In network coding, instead of simply relaying the packets they receive, the nodes of a network take several packets and combine them together for transmission. An example

![Figure 1.1: The nodes in red form a connected dominating set (CDS)](image-url)
is random linear network coding (RLNC) presented in [HMK+06]. Among its advantages is improving the network’s throughput [HKM+03]. A conclusion that can be derived from the analysis shown in [Hae11], is that RLNC spreads the information in $k$-vertex-connected graphs in $\Theta(n/k)$ rounds, with high probability. An asymptotically optimal runtime complexity according to the lower bound discussed earlier.

However, network coding requires sending large coefficients, which do not fit within the restriction on the packet size that is imposed in the Vertex-Congest model. An additional disadvantage is derived from the fact that decoding is done by solving a system of linear independent equations of $n$ variables, one variable for each of the original messages. Thus, the decoding process requires the reception of a sufficient number of packets by the node, in order to start reproducing the original information. Unfortunately, in most cases, this sufficient number of packets equals the number of original messages, which means that decoding happens only at the end of the process. This issue has supreme importance in applications of broadcasting videos or presentations. For example, when watching online content, one would prefer displaying the downloaded parts of an image immediately on the screen, rather than waiting with an empty screen until the image is fully downloaded.

Because of the disadvantages described above, it is interesting to study the behavior of store-and-forward algorithms that do not use coding. A store-and-forward algorithm is one that does not combine or alter messages (i.e. pieces of information), only stores and forwards them (perhaps with a header, depending on the packet size limit). Such a store-and-forward algorithm is described in [CHGK14a]. Almost-optimal store-and-forward algorithms that require $O(n \log(n)/k)$ rounds with high probability have been shown in [CHGK14b,CHGK14a]. These algorithms are based on a preprocessing stage which constructs (fractionally) vertex-disjoint CDS packings, which are then used in order to route messages in parallel through all the CDSs. The CDS structures are fractionally disjoint in the sense that they are weighted, and each vertex participates in several CDSs having a total weight of no more than 1. However, the failure of a single node in a CDS suffices to render the entire structure faulty. This strong reliance on the stability of the topology makes the algorithm highly sensitive to failures of nodes, an unwanted attribute when dealing with large networks. This sensitivity can be easily fixed once the structures are built by combining CDSs together into well-connected components and sending information redundantly over each CDS in the component. Combining CDSs into groups of $O(\text{polylog}(n))$ significantly improves the fault-tolerance of the algorithm, as a trade-off of a $O(\text{polylog}(n))$ factor of slowdown in runtime. Nevertheless, the construction of CDS packings in the presence of faults is still an open problem.

Randomized protocols were designed to overcome similar problems of fault-tolerance in various settings [ES09,FPRU90], as they are naturally robust and do not make decisions that depend on topological considerations. However, we show that the simple uniform random algorithm, in which every node picks and sends a message from its buffer in each round uniformly at random, is existentially slow on $k$-vertex-connected
graphs. We show that for a family of $k$-vertex-connected graphs, the uniform randomized algorithm requires $\Omega\left(\frac{n}{\sqrt{k}}\right)$ rounds in expectation to complete full information spreading.

In this research we aim to benefit from the advantages of both static and randomized routing. Our approach is to use probabilities of sending messages that change dynamically according to how the execution evolves, such that messages that are less received are more likely to be sent. This approach resembles the one in [CHG], where a fault-tolerant information spreading algorithm was designed in the GOSSIP model. However, apart from the high-level intuition, the model of communication is completely different, resulting in a completely different algorithm and analysis.

1.6 Our Contribution

We prove that the randomized algorithm, in which a node chooses the next message it sends uniformly at random from among all of the messages it has received so far, is existentially slower than optimal on $k$-vertex-connected graphs. We give a lower bound for its time complexity on $G_{n,k}$, a $k$-vertex-connected family of graphs that consist of $n/k$ cliques of size $k$ each, where nodes of each clique $C_i$ are perfectly matched to nodes of clique $C_{i-1}$, as well as nodes of clique $C_{i+1}$.

**Theorem 3.2** The uniform random algorithm requires $\Omega\left(\frac{n}{\sqrt{k}}\right)$ rounds on $G_{n,k}$, in expectation.

Next, we propose an algorithm, in which the probabilities for sending messages in each round are not fixed, but rather change dynamically during the execution based on how it evolves. Roughly speaking, the probability of sending a message is set according to the number of times it was received, with the goal of giving higher probabilities for less popular messages. The key intuition behind this approach is that nodes can take responsibility for forwarding messages that they receive few times, while they can assume that messages that have been received many times have already been forwarded throughout the network. This way, we aim to combine qualities of both random and static approaches, obtaining an algorithm that is both fast and robust. We first show the basic structure of the algorithm and prove that it is fast on $G_{n,k}$.

**Theorem 4.3** Algorithm 4.1 completes full information spreading on $G_{n,k}$ in $O\left(\frac{n}{k} \log^3 n\right)$ rounds, w.h.p.\(^2\)

The basic algorithm allows each message to be sent through multiple paths in the network, but it requires an additional mechanism in order to be robust against failures. Our next step is to augment our algorithm with some additional rounds of communication that allow the paths to change dynamically as the execution unfolds,

\(^2\)We use the phrase “with high probability” (w.h.p.) to indicate that an event happens with probability at least $1 - \frac{1}{n^c}$ for a constant $c \geq 1$. 

8
essentially bypassing faulty nodes. These shuffle phases provide fault-tolerance while retaining the round complexity of the algorithm. Here we consider a failure model in which links are reliable and nodes fail independently with probability $q$ per round, and never recover.

**Theorem 5.5** Algorithm 5.1 completes full information spreading on $G_{n,k}$ in $\tau_e = O\left(\frac{n}{k} \log^3 n\right)$ rounds, for any node failure probability per round $q$, $0 \leq q \leq \frac{1}{32\tau_e}$, w.h.p.

We view $G_{n,k}$ as an important example of a $k$-vertex-connected graph with a diameter of $\frac{n}{k}$, on which the time complexity gap between our algorithm and the uniform randomized one is conveniently analyzed. This graph emphasizes the attributes that our algorithm relies on, and we view it as a clear starting point for solving the problem for general $k$-vertex-connected graphs.

### 1.7 Thesis Roadmap

The rest of the thesis is organized as follows. In Chapter 2, we formally define the communication model and vertex-connectivity of graphs, describe the problem and the graph $G_{n,k}$, present mathematical tools widely used in our proofs, and bound the interval of relevant time complexities. In Chapter 3, we prove that the uniform random algorithm is existentially slow on $k$-vertex-connected graphs. In Chapter 4, we present the basic structure of our algorithm, and analyze its time complexity when running on $G_{n,k}$. We present an improved algorithm in Chapter 5 and show how it overcomes fault-tolerance issues of the basic algorithm. In Chapter 6, we discuss the fault-tolerance of static-routes algorithms, and possible directions for solving the problem for additional families of $k$-vertex-connected graphs.
Chapter 2

Preliminaries

We assume a network with \( n \) nodes that have unique identifiers of \( O(\log n) \) bits. Nodes perform local computations and exchange messages. We assume that local computations take negligible time relative to passing messages.

We use the phrase “with high probability” (w.h.p.) to indicate that an event happens with probability at least \( 1 - \frac{1}{n^c} \) for a constant \( c \geq 1 \). We demand our probabilistic algorithm to guarantee the spread of information during some declared time complexity with high probability.

Communication Model: In the Vertex-Congest model, each node knows its neighbors but does not know the global graph topology. The execution proceeds in a sequence of synchronous rounds. In each round, every node generates a packet and sends it to all of its neighbors. The packet size is bounded by \( O(\log(n)) \) bits and can encapsulate one message, in addition to some header.

Model of Failures: We consider a model in which links are reliable, and nodes fail independently. A node can fail at any round during the execution. A faulty node stops exchanging messages, and never recovers. We discuss two different definitions of failure probabilities. One considers the probability \( q' \) of failure of a node during the entire execution of the algorithm. That is, every node fails during the execution with probability \( q' \), and the failure might occur at any round. The second definition considers the failure probability \( q \) of a node per round. This implies smaller values of \( q \) in comparison to the values of \( q' \) in the first definition, for any fixed execution. While the first definition is more convenient when discussing a specific algorithm, we prefer to use the second definition as it is algorithm-independent and gives fair insight when comparing different algorithms with different runtimes. For example, when comparing two algorithms where one runs ten times faster than the other, it is likely that the faster algorithm faces ten times less failures, and hence it is fair to consider the rate at which nodes fail rather than the total number of failures.
**Vertex Connectivity:** An $n$-node graph is said to be $k$-vertex-connected if the graph resulting from deleting any (perhaps empty) set of fewer than $k$ vertices remains connected. In this research we assume that $k = \omega(\log^3 n)$. An equivalent definition [Men27] is that a graph is $k$-vertex-connected if for every pair of its vertices there exist $k$ vertex-disjoint paths connecting these vertices.

**Full Information Spreading:** In the problem of full information spreading, each node $u$ in the network initially holds one message of size $O(\log n)$, denoted $m_u$. The goal for each node is to collect all messages of other nodes. Namely, full information spreading is completed when every node $u$ collects messages $m_v$ for every node $v$ in the network. In the presence of failures, we still demand that all non-faulty nodes in the network receive all messages originating at faulty nodes that participated in the first round.

**The Graph $G_{n,k}$:** A key graph in our analysis is $G_{n,k}$ (see Figure 2.1). It consists of $\frac{n}{k}$ cliques, where each clique contains $k$ nodes ($n$ nodes overall). Denote by $C$ the set of all cliques. Assume an enumeration of the cliques, from $C_1$ to $C_{n/k}$. The nodes of each clique $C_i$, $i \in \{2, \ldots, \frac{n}{k}\}$, are perfectly matched with the nodes of clique $C_{i-1}$. Denote by $C(u)$ the clique that contains node $u$. A layer $L$ is a set of $n/k$ nodes from all distinct cliques that form a path starting in $C_1$ and ending in $C_{n/k}$. We denote by $\mathcal{L}$ the set of all $k$ layers. The layer $L(u) \in \mathcal{L}$ is the layer that contains node $u$. Notice that within the same clique, different nodes belong to different layers.

![Figure 2.1: Graph $G_{n,k}$](image)

### 2.1 Mathematical Background

Here we present essential inequalities used later in the analysis.
Inequalities for exponentiations

The inequalities [MV70] approximate exponentiations of $1 + x$.

**Lemma 2.1.** Consider the exponentiation $(1 + x)^r$.

- For every integer $r > 0$ and real number $x > -1$, it holds that $(1 + x)^r \geq 1 + rx$.
- For every integer $r > 0$ and real number $0 \leq x \leq 1$, it holds that $(1 - x)^r \geq 1 - rx$.
- For every integer $r > 0$ and real number $x$, it holds that $(1 + x)^r \leq e^{rx}$.

Probability bounds

The following probability bounds appear in [MU05]. Chernoff bounds are inequalities that give exponentially decreasing bounds on tail distributions of sums of independent random variables.

**Lemma 2.2** (Chernoff bounds). Let $X_1, X_2, \ldots, X_n$ be independent random variables. Let $X = \sum_{i=1}^n X_i$ and $\mu = E(X)$. Then, for $0 < \delta < 1$, it holds that:

- $ \Pr[X \leq (1 - \delta)\mu] \leq e^{-\mu \delta^2/2} ;$
- $ \Pr[X \geq (1 + \delta)\mu] \leq e^{-\mu \delta^2/3}.$

The following gives a less tight bound, which is good enough to be useful in some cases.

**Lemma 2.3** (Markov’s inequality). For a non-negative random variable $X$, and $a > 0$, it holds that

$$ \Pr[X \geq a] \leq \frac{E[X]}{a}.$$

Binomial coefficients

The following Lemma states well known bounds for $\binom{n}{k}$ [MR95].

**Lemma 2.4.** For integers $0 \leq k \leq n$, it holds that

$$ \binom{n}{k}^k \leq \binom{n}{k} \leq \left( \frac{n}{k} e \right)^k ;$$

For a fixed value of $n$, $\binom{n}{k}$ reaches its maximal value in the midpoint $k = \lfloor n/2 \rfloor$. Namely, for $1 \leq k \leq n$, it holds that

$$ \binom{n}{k} \leq \left( \frac{n}{\lfloor n/2 \rfloor} \right).$$
Repeated Bernoulli trials

A Bernoulli trial [MU05] can result in just two possible outcomes: a “success” with probability $p$, and a “failure” with probability $1 - p$.

A negative binomial experiment is a statistical experiment that has the following properties:

• The experiment consists of repeated independent and identically distributed Bernoulli trials with parameter $p$.

• A negative binomial random variable $X \sim NB(r,p)$ is the number of repeated trials to produce $r$ successes.

**Lemma 2.5.** *The expectation of a negative binomial random variable $X \sim NB(r,p)$ is*

$$E[X] = \frac{r}{p}.$$ 

A geometric random variable $X$ is the number of repeated trials with success probability $p$ to produce the first success.

**Lemma 2.6.** *The expectation of a geometric random variable $X$ with parameter $p$ is*

$$E[X] = \frac{1}{p}.$$ 

Harmonic number

The $n$-th harmonic number is the sum of the reciprocals of the first $n$ natural numbers,

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} = \sum_{i=1}^{n} \frac{1}{i}.$$ 

For any natural number $n$, the $n$-th harmonic number is bounded by natural logarithm of $n$, up to an additive factor of one [MR95].

**Lemma 2.7.** *The following holds for any natural number $n$:*

$$H_n = \sum_{i=1}^{n} \frac{1}{i} \leq \ln(n) + 1.$$ 

### 2.2 Time Complexities

Notice that for solving information spreading in the Vertex-Congest model, there is no point in re-sending messages, since the model assumes reliable links. Here we show an upper bound of $O(n)$ rounds for algorithms in which each node sends each message at most once, and does not remain quiet if it has unsent messages. Thus, the $O(n)$ bound applies for all algorithms that are reasonable in this sense. First, we prove the following, which bares some resemblance to [KLO10].
Proposition 2.8. For every round \( r \geq 0 \) and every two nodes \( u \) and \( v \) at distance \( i \), \( 0 \leq i \leq r \), at least one of the following holds:

(I) Node \( u \) sends message \( m_v \) by the end of round \( r \).

(II) Node \( u \) sends at least \( \min(r - i, n) \) different messages by the end of round \( r \).

Proof. By induction on \( i \). For each \( i \), we prove the claim for every \( r \geq i \):

**Base case** \( (i = 0) \): The distance is 0, meaning that \( u = v \). Since every node sends its own message during round 0, (I) holds for every \( r \geq i = 0 \).

**Inductive step:** Assume the proposition holds for \( i \) and every \( r \geq i \). We prove that it holds for \( i + 1 \) and every \( r \geq i + 1 \). Let nodes \( u \) and \( v \) be at distance \( i + 1 \). Let node \( u' \) be the neighbor of node \( u \) on a shortest path to \( v \). It holds that \( \text{distance}(v, u') = i \) (otherwise we get a contradiction to the definition of \( u' \), see Figure 2.2). According to the induction hypothesis, at least one of (I), (II) holds for nodes \( u' \) and \( v \). For every \( r \geq i + 1 \), we divide into two cases:

1. (I) holds for nodes \( u' \) and \( v \): \( u' \) broadcasts \( m_v \) by the end of round \( r \). Denote by \( r' \) one round before \( u' \) broadcasts \( m_v \). It is clear that \( r > r' \geq i \). The proposition holds for \( i \) and \( r' \), according to the induction hypothesis, but (I) does not hold by the choice of \( r' \). Then (II) must hold, and \( u' \) sends at least \( \min(r' - i, n) = r' - i \) different messages until the end of round \( r' \). Node \( u \) gets all these messages, with a delay of one round. It could choose to send each of them at the point of reception, if not already sent, or send another message if exists. This guarantees that \( u \) sends at least \( r' - i = (r' + 1) - (i + 1) = \min((r' + 1) - (i + 1), n) \) different messages by the end of round \( r' + 1 \), and the proposition follows for \( i + 1 \) and \( r' + 1 \).

By the choice of \( r' \), node \( u \) gets \( m_v \) from node \( u' \) no later than the end of round \( r' + 1 \). From that point, for every round \( r'' \), \( r' + 1 \leq r'' \leq r \): if \( u \) sends \( m_v \), then (I) holds and the proposition follows for \( i + 1 \) and round \( r'' \). Otherwise, node \( u \) must have chosen to send another message, that was not sent earlier, in every single round. In summary, \( u \) sends at least \( r' - i \) different messages by the end of round \( (r' + 1) \), in addition to \( r - (r' + 1) = r - r' - 1 \) messages sent later, and at least \( (r' - i) + (r - r' - 1) = r - i - 1 = r - (i + 1) \) different messages in total by the end of round \( r \). Thus, (II) holds and the proposition follows for \( i + 1 \) and round \( r \).

2. (II) holds for nodes \( u' \) and \( v \): \( u' \) sends at least \( r - i - 1 \) different messages until the end of round \( r - 1 \). Node \( u \) gets all these messages, with a delay of one round.

![Figure 2.2: A shortest path of length \( i + 1 \) between nodes \( u \) and \( v \).](image-url)
It could choose to send each of them at the point of reception, if not already sent, or send another one if exists. This guarantees that by the end of round $r$, $u$ sends at least $r - i - 1 = r - (i + 1)$ different messages. Thus, (II) holds and the proposition follows for $i + 1$ and round $r$.

This covers all possible cases.

Lemma 2.9. Upper bound: The information spreading process in our model is asymptotically bounded from above by $O(n)$ rounds.

Proof. The diameter of an $n$-node graph is bounded by $n - 1$. By Proposition 2.8, by the end of round $2n$, for every two nodes $u$ and $v$, either $u$ has sent $m_v$, or $u$ has sent at least $\min(2n - (n - 1), n) = n$ different messages. The lemma follows. □
Chapter 3

The Uniform Random Algorithm

In this chapter we consider the uniform random algorithm, in which every node picks and sends a message from its buffer in each round uniformly at random. We show that the time complexity of the algorithm is asymptotically much slower than the optimal $\Omega(n/k)$. Consider the uniform random algorithm running on graph $G_{n,k}$. We prove that the expected number of rounds for full information spreading is $\Omega(n/\sqrt{k})$. First, we prove the following.

**Lemma 3.1.** If the buffer size of a node is at least $n/4$, then the number of rounds needed for a message $m_v$ in its buffer to be sent is $\Theta(n)$ in expectation.

*Proof.* During the first $n/8$ rounds, the buffer size is at least $n/4 - n/8 = n/8$. The number of rounds until the message $m_v$ is first sent is bounded from below by a geometric random variable with success probability $p = 8/n$. By Lemma 2.6, its expectation is $1/p = n/8$, and the lemma follows. $\square$

**Theorem 3.2.** The uniform random algorithm requires $\Omega\left(\frac{n}{\sqrt{k}}\right)$ rounds on $G_{n,k}$, in expectation.

*Proof.* To prove the theorem, we define a partition over the whole space, and calculate the conditional expectations of the number of rounds for each case. We show that the expected number of rounds in every case is $\Omega(n/\sqrt{k})$, and the theorem follows according to the law of total expectation:

$$E[X] = \sum_i E[X \mid A_i] \Pr[A_i],$$

where $\{A_i\}$ is the partition.

Let $r_0$ be the random variable of the first round number in which the buffer size of all nodes in clique $C_{n/k}$ is at least $n/2$. The buffer of every node consists of the messages it knows but not sent yet. In the executions in which such round does not exist, it holds that $n - t \leq n/2$, where $t$ is the last round of the dissemination process, implying $t \geq n/2 \geq n/\sqrt{k}$. 

17
Otherwise, \( r_0 \) is well defined, and it holds that

\[
E \left[ r_0 \middle| r_0 \geq \frac{n \sqrt{k}}{32} \right] \geq \frac{n}{32 \sqrt{k}}. \tag{1}
\]

Consider the set of messages \( M_1 = \{ m_v \mid m_v \text{ is known to some node } u \in C_{n/k} \} \). We analyze two possible cases:

1. If \( |M_1| < n \) at round \( r_0 \), then there exists a message that is not known to any node in \( C_{n/k} \) at round \( r_0 \). Let \( m_v \) be such a message. Let \( r_1 \) be the random variable of the number of rounds since \( r_0 \) until the message \( m_v \) spreads to all nodes of \( C_{n/k} \). We argue that \( E[r_0 + r_1] \geq \frac{n}{32 \sqrt{k}} \). The following trivially hold, since \( r_0 \) and \( r_1 \) are non-negative:

\[
E \left[ r_0 + r_1 \middle| r_0 < \frac{n \sqrt{k}}{32}, r_1 \geq \frac{n}{24} \right] \geq \frac{n}{32 \sqrt{k}}. \tag{2}
\]

To conclude the argument for this case, it is enough to show that

\[
E \left[ r_0 + r_1 \middle| r_0 < \frac{n \sqrt{k}}{32}, r_1 < \frac{n}{24} \right] \geq \frac{n}{32 \sqrt{k}}. \tag{3}
\]

In the following, we assume that \( r_0 < n/\sqrt{k} \) and \( r_1 < n/24 \), and give a lower bound for \( E \left[ r_1 \middle| r_0 < \frac{n \sqrt{k}}{32}, r_1 < \frac{n}{24} \right] \). In addition, we assume that \( k \leq n/6 \). At round \( r_0 \) it holds that at least \( n/2 - k \geq n/3 \) messages are disseminated in \( C_{n/k-1} \), for otherwise the messages do not reach nodes of \( C_{n/k} \). During the \( r_1 \) rounds in the interval \([r_0, r_0 + r_1]\), all buffers in all nodes in \( C_{n/k} \) are of size at least \( n/2 - r_1 \), in all nodes in \( C_{n/k-1} \) are of size at least \( n/3 - r_1 \), which means that all buffer sizes of nodes both cliques are at least \( n/4 \) during the \( r_1 \) rounds in the interval. Since buffer sizes are at least \( n/4 \) and at most \( n \), the probability \( \hat{q} \) that a node \( v \in C_{n/k-1} \cup C_{n/k} \) that knows \( m_v \) sends it is \( 1/n \leq \hat{q} \leq 4/n \). Let \( X_r \) be the number of nodes in \( C_{n/k-1} \) that send \( m_v \) during the \( r \) rounds that follow round \( r_0 \). It holds that

\[
E \left[ X_r \middle| r_0 < \frac{n \sqrt{k}}{32}, r_1 < \frac{n}{24} \right] \text{ is at most } k(1-(1-\hat{q})^r) \leq k(1-(1-\hat{q}r)) = k\hat{q}r \leq \frac{4kr}{n}
\]

(the first inequality is according to Lemma 2.1). Each node in \( C_{n/k-1} \) that sends the message relays it to its corresponding neighbor in \( C_{n/k} \). If any of these nodes in \( C_{n/k} \) sends \( m_v \), then the message is disseminated and all \( k \) nodes in \( C_{n/k} \) know it.

For every \( r \geq n/k \), denote by \( A_r \) the event \( X_r \leq \frac{8kr}{n} \). By applying Markov’s inequality we get that \( \Pr \left[ X_r \geq \frac{8kr}{n} \middle| r_0 < \frac{n \sqrt{k}}{32}, r_1 < \frac{n}{24} \right] \leq \frac{4kr}{n} / \frac{8kr}{n} = \frac{1}{2} \), and hence,

\[
\Pr \left[ A_r \middle| r_0 < \frac{n \sqrt{k}}{32}, r_1 < \frac{n}{24} \right] \geq \frac{1}{2}.
\]

Under the assumption that \( A_r \) occurs, the probability for the dissemination to occur during these \( r \) rounds (implying that \( r_1 \leq r \)) is at most

\[
1 - ((1-\hat{q})^r)^{8kr/n} = 1 - (1-\hat{q})^{8kr^2/n} \leq 1 - (1-8kr^2\hat{q}/n) = 8kr^2\hat{q}/n \leq \frac{32kr^2}{n^2},
\]

(first inequality is
according to Lemma 2.1). By assigning \( r = \frac{n}{8\sqrt{k}} > n/k \), we get that

\[
\Pr \left[ r_1 > \frac{n}{8\sqrt{k}} \bigg| A_r, r_0 < \frac{n}{\sqrt{k}}, r_1 < \frac{n}{24} \right] \geq 1 - \frac{32k(n/8\sqrt{k})^2}{n^2} \geq \frac{1}{2},
\]

and hence

\[
\Pr \left[ r_1 > \frac{n}{8\sqrt{k}} \bigg| r_0 < \frac{n}{\sqrt{k}}, r_1 < \frac{n}{24} \right] \geq \frac{1}{2},
\]

the first equality is according to the law of conditional probability, \( P(A \cap B) = P(A|B)P(B) \). This gives

\[
E \left[ r_1 \bigg| r_0 < \frac{n}{\sqrt{k}}, r_1 < \frac{n}{24} \right] \geq \frac{n}{32\sqrt{k}},
\]

which proves (3).

2. If \(|M_1| = n\) at \( r_0 \), let \( r_2 \) be the random variable of the number of rounds since \( r_0 \) until all messages are disseminated in \( C_{n/k} \). We argue that \( E[r_0 + r_2] \geq \frac{n}{32\sqrt{k}} \).

Since \( r_0 \) and \( r_2 \) are non-negative, the following hold trivially:

\[
E \left[ r_0 + r_2 \bigg| r_0 < \frac{n}{\sqrt{k}}, r_2 \geq \frac{n}{24} \right] \geq \frac{n}{32\sqrt{k}}, \tag{4}
\]

To conclude the argument for this case, it is enough to show that

\[
E \left[ r_0 + r_2 \bigg| r_0 < \frac{n}{\sqrt{k}}, r_2 < \frac{n}{24} \right] \geq \frac{n}{32\sqrt{k}}. \tag{5}
\]

In the following, we assume that \( r_0 < n/\sqrt{k} \) and \( r_2 < n/24 \), and give a lower bound for \( E \left[ r_2 \bigg| r_0 < \frac{n}{\sqrt{k}}, r_2 < \frac{n}{24} \right] \). In addition, we assume that \( k \leq n/6 \). In each round, a node in \( C_{n/k} \) receives at most \( k \) new messages and sends one, and hence the buffer size can increase by at most \( k - 1 \) in a single round. By its definition, at round \( r_0 \) there exists a node in \( C_{n/k} \) with buffer size at most \( n/2 + k - 1 \leq 2n/3 \), implying that the number of disseminated messages in \( C_{n/k} \) is at most \( n/2 + r_0 + k - 1 \leq n/4 \). At round \( r_0 \), at least \( n/4 \) messages are not disseminated in \( C_{n/k} \) but are known to some nodes of the clique. Denote the set of these messages by \( M_2 \). Messages in \( M_2 \) were not received from nodes within the clique \( C_{n/k} \) (otherwise, there are disseminated), which means that at round \( r_0 \) every node in \( C_{n/k} \) knows at most \( r_0 < n/\sqrt{k} \) such messages. During the
$r_2$ rounds in the interval $[r_0, r_0 + r_2]$, all buffers in all nodes in $C_{n/k}$ are of size at least \( n/2 - r_2 \geq n/2 - n/\sqrt{k} - n/24 \geq n/4 \). Since buffer sizes are at least $n/4$, the probability $\hat{q}$ for each node in $C_{n/k}$ to send a message from $M_2$ in a single round is at most $\frac{n}{\sqrt{k}}/\frac{n}{4} = \frac{1}{\sqrt{k}}$. In order for the dissemination process to complete, each message in $M_2$ must be sent at least once by some node in $C_{n/k}$ (or be sent by all nodes of $C_{n/k-1}$, which happens only after $\Omega(n)$ rounds in expectation, by Lemma 3.1). By considering the sending of a message from $M_2$ a success, which occurs with probability at most $\hat{q}$, the dissemination process completes after at least $|M_2|/\hat{q}$ successes. Denote by $X$ the random variable of number of trials before reaching $|M_2|$ successes. $X \sim NB(|M_2|, \hat{q})$, with expectation of $|M_2|/\hat{q} \geq \frac{n}{4}/\frac{1}{\sqrt{k}} = n\sqrt{k}$ trials (by Lemma 2.5). In each round, the number of trials is $k$ (one trial per node of the clique), and hence, the expected number $r_2$ of additional rounds before all messages are disseminated in $C_{n/k}$ is at least $n\sqrt{k}/k = \frac{n}{\sqrt{k}}$ in expectation. We get that

$$E \left[ r_2 \mid r_0 < \frac{n}{\sqrt{k}}, r_2 < \frac{n}{24} \right] \geq \frac{n}{\sqrt{k}},$$

which proves (5).

In summary, we covered the whole space by combinations of events that form a partition, proved that the conditional expectation in each case is $\Omega(n/\sqrt{k})$, and hence by the law of total expectation, the theorem follows. \qed
Chapter 4

A Fast Information Spreading Algorithm

In the following section we describe our basic information spreading algorithm. We emphasize that the algorithm does not assume anything about the underlying graph, except for a polynomial bound on its size. In particular, the nodes do not know the vertex-connectivity of the graph, nor any additional information about its topology.

4.1 Algorithm Description

Each node $u$ has a buffer of received messages, whose content at the beginning of round $t$ is denoted $R_u(t)$. We use $cnt_{u,v}(t)$ to denote the number of times a node $u$ has received message $m_v$ by the beginning of round $t$. Denote by $S_u(t)$ the set of messages sent by node $u$ by the beginning of round $t$. Define $B_u(t) \equiv R_u(t) - S_u(t)$, the set of messages that are known to node $u$ at the beginning of round $t$, but not yet sent. We refer to $B_u(t)$ as a logical variable, whose value changes implicitly according to updates in the actual variables $R_u(t)$ and $S_u(t)$. For every node $u$, we have that $S_u(0) = \emptyset$, $R_u(0) = \{m_u\}$, $cnt_{u,u}(0) = 1$, and for each $v \neq u$, $cnt_{u,v}(0) = 0$.

We present an algorithm, Algorithm 4.1, that consists of two types of phases: a random phase and ranking phases. Let $t_0$ be the round number at the beginning of the random phase, and let $\tilde{t}_0$ be the round number after the random phase. Let $t_p$ be the round number at the beginning of ranking phase $p$, and let $\tilde{t}_p$ be the round number after ranking phase $p$. Denote by $\tilde{B}_u(t_p)$ the buffer of node $u$ at time $t_p$. Unlike $B_u(t)$, $\tilde{B}_u(t)$ is an actual variable that does not implicitly change according to $R_u(t)$ and $S_u(t)$. We assign a value to it at the beginning of every phase, that is, $\tilde{B}_u(t_p) = B_u(t_p)$, and make sure that its content only gets smaller during a phase.

In this algorithm, it holds that $\tilde{t}_p = t_{p+1}$ for every $p$, and $t_0 = 1$. We will later modify this algorithm in Chapter 5, where we argue on properties that hold in $\tilde{t}_p$ and $t_{p+1}$, separately. The parameters $\alpha$ and $d$ are constants that are fixed later, at the end of Section 4.2. The algorithm runs as follows (see Figure 4.1), where in each round
every node sends a message and receives messages from all of its neighbors:

(1) Single round (Round 0): This is the first round of the algorithm, where every node $u$ sends the message $m_u$ it has.

(2) Random phase: This is the first phase of the algorithm, which consists of $\tau = \alpha \log n$ rounds. In each round $t$, every node $u$ picks (and pops) a message from buffer $\hat{B}_u(t_0)$ uniformly at random.

(3) Consecutive ranking phases: Each of these phases consists of $\tau' = 8d\tau \log^2 n$ rounds. At the beginning of such a phase, each node uses the Ranking Function (Figure 4.2) that defines a probability space over the messages in $\hat{B}_u(t_p)$. In each round, every node $u$ picks (and pops) a message to send from buffer $\hat{B}_u(t_p)$ according to the probability space.

Next, we present the probability space used in ranking phases:

**Ranking Function:** The ranking function (in Figure 4.2) is calculated by each node, and defines a probability space over its messages. Each node $u$ sorts the messages in buffer $\hat{B}_u$ according to their $cnt$ values, smallest to largest, breaking ties arbitrarily. Denote by $rank_m$ the position of the message $m$ within the sorted list, and let $b = |\hat{B}_u|$, be the size of the list. We consider the probability space in which the probability for a message $m$ with $rank_m = r$ to be picked is

$$\frac{1}{r \cdot H_b}.$$  

Namely, the probability is inversely proportional to $r$. The $b$-th harmonic number, $H_b = \sum_{i=1}^{b} \frac{1}{i}$, is a normalization factor (over the whole list of messages). This mean that messages in lower positions (lower $rank_m$ values, implying lower $cnt$ values) are more likely to be picked.

In the algorithm, the probability space used by a node $u$ during a phase is calculated at the start of the phase. In ranking phases, it is defined according to the Ranking function. In the random phase, it is the uniform distribution. Within a phase, the
only modifications in the probability space of a node are done due to the non-repetitive sending policy, i.e., the need for nullifying probabilities of messages that are already sent. When a message is sent, the modification can be done, for example, by updating the normalization factor, or alternatively by distributing the probability of the sent message between all other messages (say, proportionally to their current probabilities). Anyhow, this implies that the probability of each message can only get larger during a phase, as long as it is not sent. Namely, the initial probability of a message (at the beginning of a phase) is a lower bound on its probability for the rest of the phase (as long as it is not sent). Probabilities are not defined for messages that were not known at the start of a phase, and were first received during the phase, thus these messages have no chance of being sent until the next phase starts.

Algorithm 4.1 for each node $u$

1: SyncRound($m_u$) \hfill $\triangleright$ Round 0
2: RandomPhase()
3: loop
4: \hspace{1em} RankingPhase()
5: end loop

SyncRound($m$)

6: procedure SyncRound($m$) \hfill $\triangleright$ A synchronized round
7: \hspace{1em} send($m$)
8: \hspace{1em} $S_u(t) \leftarrow S_u(t) \cup \{m\}$
9: \hspace{1em} $R \leftarrow$ received messages
10: \hspace{1em} for all $m_v \in R$ do
11: \hspace{2em} $R_u(t) \leftarrow R_u(t) \cup \{m_v\}$
12: \hspace{2em} $\text{cnt}_{u,v}(t) \leftarrow \text{cnt}_{u,v}(t) + 1$
13: \hspace{1em} end for
14: \hspace{1em} $t \leftarrow t + 1$
15: end procedure

RandomPhase

16: \hspace{1em} $\hat{B}_u(t_0) \leftarrow B_u(t)$ \hfill $\triangleright$ $t = t_0$
17: \hspace{1em} loop $\tau$ times \hfill $\triangleright$ $\tau = \alpha \log n$
18: \hspace{2em} $m \leftarrow$ pop message from $\hat{B}_u(t_0)$ uniformly at random
19: \hspace{2em} SyncRound($m$)
20: end loop

RankingPhase $p$

21: \hspace{1em} $\hat{B}_u(t_p) \leftarrow B_u(t)$ \hfill $\triangleright$ $t = t_p$
22: \hspace{1em} Prob $\leftarrow$ RANKINGFUNCTION($\hat{B}_u(t_p)$)
23: \hspace{1em} loop $\tau'$ times \hfill $\triangleright$ $\tau' = 8d\tau \log^2 n$
24: \hspace{2em} $m \leftarrow$ pop message from $\hat{B}_u(t_p)$ according to Prob
25: \hspace{2em} Nullify Prob[$m$] (update Prob accordingly)
26: \hspace{2em} SyncRound($m$)
27: end loop
The Phase Separation Property. Changes in cnt values during a phase (due to reception of messages) do not affect the probability space of this phase, as it is calculated only at the start of each phase. This implies that messages that are first received by a node after the start of the random phase or a ranking phase have zero probability for being sent during that phase, and can be sent by the node only starting from the next phase, when the probability space is recalculated. We call this the phase separation property, and it implies the following:

Proposition 4.1. At the start of ranking phase $p$, every message has propagated to a distance of at most $p + 1$.

The following lemma holds for any node and for a general graph:

Lemma 4.2. Let $m$ be a message with rank $r \leq 8\tau$, then $m$ is sent during the ranking phase with probability at least $1 - \frac{1}{n_d}$.

Proof. Let $A$ be the event that the message with rank $r$ is not picked during a phase of $\tau' = 8d\tau \log^2 n = 8d\alpha \log^3 n$ rounds. We wish to bound from above the probability for event $A$:

\[
\Pr[A] \leq \left(1 - \frac{1}{r \cdot H_b}\right)^{\tau'} \leq \left(1 - \frac{1}{r \cdot (\ln b + 1)}\right)^{\tau'} \leq \left(1 - \frac{1}{r \cdot \log b}\right)^{\tau'} \leq \left(1 - \frac{1}{r \log n}\right)^{8d\alpha \log^3 n} \leq \left(1 - \frac{1}{r \log n}\right)^{8d\alpha \log^2 n} \leq \left(1 - \frac{1}{\log n}\right)^{\frac{1}{2}8d\alpha \log^2 n} = \left(\frac{1}{n}\right)^{\frac{1}{2}8d\alpha \log n}.
\]

The second inequality holds according to Lemma 2.7. The last inequality holds according to $(1 - 1/x)^x \leq e^{-1} < 1/2$ for $x > 0$. Namely, any message with $r \leq 8\tau = 8\alpha \log n$ is sent during the phase with probability at least $1 - \frac{1}{n_d}$. \hfill \Box

4.2 Time Analysis for $G_{n,k}$

Recall that $G_{n,k}$ is the graph that consists of $n/k$ cliques of size $k$ (assume $n/k$ is an integer), with a matching between every two consecutive cliques (see Figure 2.1). Clearly, $G_{n,k}$ is $k$-vertex-connected.
We now analyze the time complexity of the algorithm to spread information over $G_{n,k}$. For simplicity, we analyze the flow of messages from $C_j$ to $C_i$, where $j \leq i$. The opposite direction of flow and its analysis are symmetric.

**Theorem 4.3.** Algorithm 4.1 completes full information spreading on $G_{n,k}$ in $O\left(\frac{n}{k} \log^3 n\right)$ rounds, w.h.p.

To prove the theorem, we prove the following:

**Lemma 4.4 (Iteration).** For every $i, 1 \leq i \leq \frac{n}{k}$, every node $u \in C_i$, and every node $v$ such that $v \in C_j$ for some $i - p \leq j \leq i$, it holds that $m_v \in R_u(t_p)$, w.h.p.

This directly proves Theorem 4.3, as follows.

**Proof of Theorem 4.3.** Lemma 4.4 shows that by the end of ranking phase $p$, w.h.p. each node $u$ knows all messages $m_v$ originating at distance at most $p$. This implies that full information spreading is completed after $n/k$ phases, since $n/k$ is the diameter of the graph, which proves Theorem 4.3. $\square$

The following definition is useful to indicate that a node shares responsibility for disseminating a message.

**Definition 4.5.** A fresh message of a node $u$ at time $t$, is a message $m_v \in R_u(t)$ for which $\text{cnt}_{u,v}(t) < T$, where $T$ is set to be $\frac{1}{2} \tau$.

**General Idea of the Proof.** At the end of round 0, every message $m_v$ is disseminated in its own clique $C(v)$. Then, we show that by the end of the random phase, each message $m_v$ is sent w.h.p. by a sufficiently large number of nodes $u \in C(v)$, to become non-fresh in all nodes of the clique $C(v)$. Simultaneously, each of the messages $m_v$ becomes known and fresh in a sufficiently large number of nodes in the neighboring clique.

Then we show that ranking phases shift and preserve this situation. At the beginning of every ranking phase, every fresh message in a node is also fresh in a sufficiently large number of nodes within the same clique. During the phase, all of the fresh messages are sent w.h.p., implying that each one of the messages (i) is disseminated in the clique; (ii) is not fresh in nodes of the clique anymore; and (iii) is fresh in a sufficiently large number of nodes in the neighboring clique.

The combination of properties (ii) and (iii) is the crux of the proof. It guarantees that the process progresses iteratively, as it leads to similar conditions again and again at the beginning of every new ranking phase. This happens because every node can easily distinguish between a new message received from nodes within the clique (becomes non-fresh by the end of the phase), and a new message received from the neighbor in the neighboring clique (stays fresh at the end of the phase, and should be sent during
the next phase). We emphasize that this is all done implicitly, and that the nodes do not know the structure of the network.

This iterative behavior of the combined properties guarantees that every message propagates one additional clique per phase, until full information spreading completes after $O(n/k)$ phases. In Figure 4.3, see an illustration of the expected routes on which a message originated at $C_1$ is relayed to reach $C_{n/k}$.

![Figure 4.3: Illustration for expected routes of a message from $C_1$ to $C_{n/k}$ in Algorithm 4.1](image)

Let $t'$, for $0 \leq t' \leq \tau - 1$, be the time from the first round of the random phase, i.e., $t' = t - t_0$. The following proposition is immediate from the pseudocode:

**Proposition 4.6.** At the beginning of the random phase, $\hat{B}_u(t_0)$ for every node $u \in C_i$ contains exactly $k - 1$ messages $m_v$ originating at $v \in C_i$, and at most two additional messages, one originating at $v \in C_{i-1} \cap L(u)$, and one originating at $v \in C_{i+1} \cap L(u)$. Thus, it holds that $|\hat{B}_u(t_0 + t')| = k + 1 - t'$, for $i = 2, 3, \ldots, \frac{n}{k} - 1$, and $|\hat{B}_u(t_0 + t')| = k - t'$, for $i = 1, \frac{n}{k}$.

Namely, nodes of inner cliques ($C_i, 1 < i < n/k$) start the random phase with $|\hat{B}_u(t_0)| = k + 1$, while nodes of cliques $C_1$ and $C_{n/k}$ start the random phase with $|\hat{B}_u(t_0)| = k$.

### 4.2.1 Analysis of the Random Phase

The following lemma analyzes the initial random phase, and shows that every message $m_v$ is non-fresh in all nodes of $C(v)$ at the end of the random phase:

**Lemma 4.7.** At the end of the random phase, for every message $m_v$ and for all nodes $u \in C(v)$, $m_v$ is non-fresh for $u$, with probability at least $1 - \frac{1}{n^{\alpha/48 - 1}}$.

**Proof.** Fix $v$. Message $m_v$ is disseminated in $C(v)$ by the start of the random phase. By Proposition 4.6, for every $u \in C(v)$, it holds that $|\hat{B}_u(t_0 + t')| \leq k + 1 - t'$ during the random phase.

26
Let $\mathbb{1}_{u,v}$, for every $u \in C(v)$, be an indicator variable that indicates whether node $u$ sends $m_v$ during the random phase or not. Then

$$\Pr[\mathbb{1}_{u,v} = 1] \geq 1 - \prod_{t'=0}^{t-1} \frac{k - t'}{k + 1 - t'} = 1 - \frac{k + 1 - \tau}{k + 1} \geq \frac{\tau}{(3/2)k}.$$ 

Let $X_v = \sum_{u \in C(v)} \mathbb{1}_{u,v}$, be the number of nodes in $C(v)$ that send $m_v$ during the random phase, i.e., the number of times $m_v$ is received by every node in $C(v)$. Then

$$\mu = E(X_v) = E\left( \sum_{u \in C(v)} \mathbb{1}_{u,v} \right) = \sum_{u \in C(v)} E(\mathbb{1}_{u,v}) \geq \sum_{u \in C(v)} \frac{2\tau}{3k} = \frac{2\tau}{3}.$$

Since $v$ is fixed, the indicator variables are independent, as they refer to decisions of distinct nodes. By applying a Chernoff bound (Lemma 2.2), we get

$$\Pr[X_v \leq (1 - \delta)\mu] \leq \exp\left(-\delta^2 \frac{\mu}{2}\right) \leq \exp\left(-\delta^2 \frac{\alpha \log n}{3}\right) < \frac{1}{n^{\frac{\alpha}{4}}}.$$

By setting $\delta = \frac{1}{4}$, we get that a message $m_v$ is non-fresh in all nodes $u \in C(v)$ with probability at least $1 - \frac{1}{n^{\alpha/16}}$. By a union bound, this holds for every node $v$ with probability at least $1 - \frac{1}{n^{\alpha/24}}$.

Next, we consider messages that are fresh.

**Definition 4.8.** A pioneer message in node $u \in C_i$ at time $t_p$ (beginning of ranking phase $p$), is a message $m_v \in R_u(t_p)$ that originated at $v \in C_{i-p-1}$.

**Pioneer Attributes.** If a message $m_v$ is a pioneer in node $u \in C_i$ at time $t_p$, then (i) $v \in L(u)$ (by Proposition 4.1, the message was transmitted over the shortest path), and the following hold at time $t_p$: (ii) $\text{cnt}_{u,v}(t_p) = 1$, and thus $m_v$ is fresh for $u$, (iii) $m_v \notin R_{u'}(t_p)$ for every $u' \in C_i, u' \neq u$ (by Proposition 4.1), (iv) $m_v$ is disseminated in $C_{i-1}$ (by the node that relayed $m_v$ to its neighbor in $C_i$), and (v) $m_v$ is fresh in every node $u' \in C_{i-1}$.

**Lemma 4.9.** With probability at least $1 - \frac{1}{n^{\alpha/24}}$, at the end of the random phase, for every $i$, the number of pioneer messages that reach $C_i$ is at most $3\tau$.

**Proof.** According to pioneer definition, considering the direction of the flow of messages, cliques $C_1$ and $C_2$ could not have pioneer messages. Fix $i$, $3 \leq i \leq n/k$. By Proposition 4.6, at the beginning of the random phase, for every node $u \in C_{i-1}$, buffer $\hat{B}_u(t_0)$ contains exactly one unique message $m_v, v \in C_{i-2} \cap L(u)$, and it holds that $|\hat{B}_u(t_0 + t')| = k + 1 - t'$ during the random phase (as $C_{i-1}$ is an inner clique).

Let $\mathbb{1}_u$, for every $u \in C_{i-1}$, be an indicator variable that indicates whether node $u$
sends its unique message during the random phase, or not. Then

\[ \Pr[\mathbb{1}_u = 1] = 1 - Pr[\mathbb{1}_u = 0] = 1 - \prod_{t=0}^{\tau-1} \frac{k - t'}{k + 1 - t'} = 1 - \frac{k + 1 - \tau}{k + 1} = 1 - \left(1 - \frac{\tau}{k + 1}\right) = \frac{\tau}{k + 1}. \]

Let \( X_{i-1} = \sum_{u \in C_{i-1}} \mathbb{1}_u \), be the number of messages \( m_v, v \in C_{i-2} \), that reach clique \( C_i \) by the end of the random phase. Then

\[ \mu = E(X_{i-1}) = E\left( \sum_{u \in C_{i-1}} \mathbb{1}_u \right) = \sum_{u \in C_{i-1}} E(\mathbb{1}_u) = \sum_{u \in C_{i-1}} \frac{\tau}{k + 1} = k \cdot \frac{\tau}{k + 1}, \]

which means that \( \tau/2 \leq \mu \leq \tau \). The indicator variables are independent, as they refer to decisions of distinct nodes. By applying a Chernoff bound (Lemma 2.2), we get

\[ \Pr[X_{i-1} > (3/2)\tau] \leq \Pr[X_{i-1} \geq (3/2)\mu] \leq \Pr[X_{i-1} \geq (1 + \delta)\mu] \leq \exp\left(\frac{-\delta^2 \cdot \mu}{3}\right) \leq \exp\left(\frac{-\delta^2 \cdot (\tau/2)}{3}\right) \leq \exp\left(\frac{-\delta^2 \cdot \alpha \log n}{6}\right) < \frac{1}{n^{a \delta^2}}. \]

By setting \( \delta = \frac{1}{2} \), we get that the number of pioneer messages, \( X_{i-1} \), that reach \( C_i \) from one direction is \( \leq (3/2)\tau \) with probability at least \( 1 - \frac{1}{n^{a \delta^2}} \). By a union bound, this holds for both directions and every clique with probability at least \( 1 - \frac{1}{n^{a \delta^2}} \). \( \square \)

### 4.2.2 Analysis of Ranking Phases

After analyzing the single random phase, here we analyze the ranking phases.

**Lemma 4.10.** With probability at least \( 1 - \frac{1}{n^{a \delta^2}} \), every node \( u \) that starts ranking phase \( p \) with at most \( 8\tau \) fresh messages, sends all of them during the phase.

**Proof.** Fix a node \( u \). All fresh messages \( m_v \in R_u(t) \) have rank \( r \leq 8\tau \). According to Lemma 4.2, a message with rank \( r \leq 8\tau \) is sent during a ranking phase with probability at least \( 1 - \frac{1}{n^{a \delta^2}} \). By a union bound, the probability for node \( u \) to send all of its fresh messages during the phase is bounded by \( 1 - \frac{1}{n^{a \delta^2}} \cdot 8\tau \geq 1 - \frac{1}{n^{a \delta^2}} \). We use a union bound once more to bound the probability that this happens for every node \( u \) by \( 1 - \frac{1}{n^{a \delta^2}} \cdot n = 1 - \frac{1}{n^{a \delta^2}} \). \( \square \)

To prove Lemma 4.4, we show a sequence of four inductive properties, that hold for ranking phase \( p \), with probability at least

\[ 1 - \left(\frac{2p}{n^{a \delta^2}} + \frac{2}{n^{a \delta^2}}\right). \]
Property 1. For every $i, 1 \leq i \leq \frac{n}{k}$, it holds that the number of messages $m_v, v \in C_{i-p-1}$, such that $m_v \in R_u(t_p)$ for some $u \in C_i$ (pioneers), is at most $3\tau$, and each reaches a distinct node $u \in L(v)$.

Property 2. For every $i, 1 \leq i \leq \frac{n}{k}$, and every node $u \in C_i$, it holds that at time $t_p$ there are at most $4\tau$ fresh messages $m_v$ for node $u$ for every one of the two directions of flow ($8\tau$ in total). All of them originated at nodes $v \in C_{i-p}$ (similarly, $v \in C_{i+p}$), except for at most one (a pioneer) which originated at $u' \in C_{i-p-1} \cap L(u)$ (similarly, $u' \in C_{i+p+1} \cap L(u)$). All messages $m_v \in R_u(t_p), v \in C_{i-p}$ (similarly, $v \in C_{i+p}$), are fresh.

Property 3. For every $i, 1 \leq i \leq \frac{n}{k}$, and every node $v \in C_{i-p}$, it holds that $m_v$ is fresh for at least $T$ nodes $u \in C_i$ at time $t_p$. Recall that $T = \frac{1}{2}\tau$.

Property 4. For every $i, 1 \leq i \leq \frac{n}{k}$, every node $u \in C_i$, and every node $v$ such that $v \in C_j$ for some $i - p \leq j \leq i$, it holds that $m_v \in R_u(t'_p)$, and $m_v$ is non-fresh.

We prove the four properties simultaneously by induction on the ranking phase number, $p$. To prove the base cases, we assume that all events described in Lemma 4.7, Lemma 4.9, and Lemma 4.10 (for $p = 1$) occur. Notice that, by a union bound, the probability for this is at least $1 - \left(\frac{1}{n^{\alpha/2k-\tau}} + \frac{1}{n^{\alpha/4k-\tau}} + \frac{1}{n^{d/2}}\right) \geq 1 - \left(\frac{2}{n^{\alpha/2k-\tau}} + \frac{2}{n^{d/2}}\right)$.

Base case for Property 1. Let $p = 1$. One random phase precedes the first ranking phase. The upper bound on the number of pioneers in every clique holds according to Lemma 4.9. The distribution among distinct layers is immediate according to Attribute (i) of pioneer messages.

Base case for Property 2. Let $p = 1$. Fix some node $u \in C_i$. We analyze possibilities for fresh messages for one direction of flow at the end of the random phase, and the other direction is symmetric. By Proposition 4.1, messages $m_v \in R_u(t_1)$ originate at nodes $v \in C_{i-2} \cup C_{i-1} \cup C_i$. A message $m_v \in R_u(t_1)$ that originates at node $v \in C_{i-2}$ is a pioneer. By Attributes (i) and (ii) there can be at most one such message, and it is fresh. For messages $m_v \in R_u(t_1)$ that originate at nodes $v \in C_{i-1}$ there are two possibilities. One possibility is that they are received from the neighbor $u' \in C_i \cap L(v)$, which implies that they are pioneers in nodes $u_1 \in C_{i+1} \cap L(v)$ at time $t_1$. By Property 1 for $p = 1$ (which is already proved), there are at most $3\tau$ such messages. The only other possibility is that they are received from the neighbor $u' \in C_{i-1} \cap L(u)$. There are at most $\tau$ such messages (which might include one that originates at $C_{i-2}$, as already discussed), and they are all fresh. Messages $m_v \in R_u(t_1)$ that originate at nodes $v \in C_i$ are all non-fresh, according to Lemma 4.7.

In total, at the beginning of the first ranking phase, each node $u$ has at most $4\tau$ fresh messages from the one direction. All of them originated at nodes $u' \in C_{i-1}$, except
for at most one which originated at \( u' \in C_{i-2} \cap L(u) \). All messages that originated at nodes \( u' \in C_{i-1} \) are fresh. The other direction of flow is symmetric.

**Base case for Property 3.** Let \( p = 1 \). For every \( v \in C_{i-1} \), at the end of round 0, exactly one node \( u \in C_i \) knows \( m_v \). It may disseminate it during the random phase. At the end of the random phase, by Lemma 4.7, for every \( v \in C_{i-1} \), \( m_v \) is non-fresh in all nodes of \( C_{i-1} \). That is, by the end of the random phase, every node \( v' \in C_{i-1} \), \( v' \neq v \), receives \( m_v \) at least \( T \) times, all from nodes within the clique. Therefore, at least \( T \) nodes in \( C_{i-1} \) send \( m_v \) in the random phase, which implies that at least \( T \) nodes in \( C_i \) know \( m_v \). According to the phase separation property, every such node in \( C_i \) receives \( m_v \) at most twice (from the neighbor in \( C_{i-1} \), and possibly from the neighbor \( u \in C_i \)), so it is fresh.

**Base case for Property 4.** Let \( p = 1 \). Fix \( i, u \in C_i \). According to Lemma 4.7, it holds that for every node \( v \in C_i \), \( m_v \) is known and non-fresh in \( u \).

At the beginning of the first ranking phase, according to Property 3 for \( p = 1 \) and \( i \), it holds that every message \( m_v, v \in C_{i-1} \), is fresh in at least \( T \) nodes in \( C_i \). According to Property 2 for \( p = 1 \), it holds that every node has at most 8\( \tau \) fresh messages. By Lemma 4.10, all nodes (in particular, nodes in \( C_i \)) send all of their fresh messages. This means that every message \( m_v, v \in C_{i-1} \), is received by node \( u \) at least \( T \) times so it becomes non-fresh.

This completes the proof of the base cases. Recall that the base cases are proved by assuming that all events described in Lemma 4.7, Lemma 4.9, and Lemma 4.10 (for \( p = 1 \)) occur. Thus, the properties are proved for \( p = 1 \) with probability at least \( 1 - \left( \frac{2}{n^{\alpha/48-1}} + \frac{2}{n^{\alpha/2-2}} \right) \).

To prove the induction step, we assume that all events described in the four properties for \( p - 1 \), and in Lemma 4.10 for \( p - 1 \) and \( p \), occur. This happens with probability at least \( 1 - \left( \frac{2}{n^{\alpha/48-1}} + \frac{2(p-1)}{n^{\alpha/2-2}} + \frac{1}{n^{\alpha/2}} + \frac{1}{n^{\alpha/2}} \right) = 1 - \left( \frac{2}{n^{\alpha/48-1}} + \frac{2p}{n^{\alpha/2}} \right) \).

**Induction step for Property 1.** By Property 1 for \( p - 1 \) and \( i - 1 \), at the beginning of ranking phase \( p - 1 \), the number of messages \( m_v, v \in C_{i-p-1} \), that reach nodes in \( C_{i-1} \) is at most \( 3\tau \), each reaches a distinct node \( u \in C_{i-1} \cap L(v) \). At time \( t_{p-1} \), by Pioneer Attribute (ii), each one of them is fresh. By Property 2 for \( p - 1 \) and \( i - 1 \), at the beginning of ranking phase \( p - 1 \), every node \( u \in C_{i-1} \) has at most \( 8\tau \) fresh messages. By Lemma 4.10 for \( p - 1 \), every node sends all of its fresh messages during ranking phase \( p - 1 \) (in particular, pioneer messages in nodes in \( C_{i-1} \)). Thus, it holds that the number of messages \( m_v, v \in C_{i-p-1} \), such that \( m_v \) is a pioneer at time \( t_p \) in nodes of \( C_i \), is at most \( 3\tau \), and each reaches a distinct node \( u \in C_i \cap L(v) \).
Induction step for Property 2. Fix a node $u \in C_i$. By Proposition 4.1, for every message $m_v \in R_u(t_p)$ (known to $u$ at the beginning of ranking phase $p$) it holds that $v \in \bigcup_{j \in \{1, \ldots, i\}} C_j$. By Property 4 for $p - 1$ and $i$, for every node $v$ such that $v \in C_j$ for some $i - p + 1 \leq j \leq i$, it holds that $m_v \in R_u(\bar{t}_{p-1})$, $m_v$ non-fresh. Thus, only messages $m_v, v \in C_{i-p-1} \cup C_{i-p}$ can be fresh.

Consider a message $m_v, v \in C_{i-p}$: By Property 1 for $p - 1$ and $i$, at the beginning of ranking phase $p - 1$, any message $m_v, v \in C_{i-p}$, that reach $C_i$ (a pioneer) is known to exactly one node in the clique. Thus, any message $m_v, v \in C_{i-p}$, that reaches $C_i$ by the beginning of ranking phase $p$ is fresh (because it could be received only once from a neighbor within the clique $C_i$ and once from a neighbor in clique $C_{i-1}$, i.e., it is received at most twice).

By Property 2 for $p - 1$ and $i - 1$, at the beginning of ranking phase $p - 1$, node $u' \in C_{i-1} \cap L(u)$ has at most $4\tau$ fresh messages (consider relevant direction of flow), all of them originated at nodes $v \in C_{i-p}$, except for at most one which originated at $u' \in C_{i-p-1} \cap L(u)$ (a pioneer). According to Lemma 4.10 for $p - 1$, every node $u'$ sends all of its fresh messages during ranking phase $p - 1$. Thus, at the end of ranking phase $p - 1$ (beginning of ranking phase $p$), they all reach $u$, and they are all fresh. In particular, they are received at most twice, according to the previous discussion. The opposite direction of flow is symmetric. This completes the proof. □

Induction step for Property 3. By Property 3 for $p - 1$ and $i - 1$, at the beginning of ranking phase $p - 1$, every message $m_v, v \in C_{i-p}$, is fresh in at least $T$ nodes $u' \in C_{i-1}$. By Property 2 for $p - 1$ and $i - 1$, at the beginning of ranking phase $p - 1$, every node $u \in C_{i-1}$ has at most $8\tau$ fresh messages. According to Lemma 4.10 for $p - 1$, all are sent during ranking phase $p - 1$, each of the nodes $u \in C_{i-1}$ sends to a distinct neighbor node $u \in C_i$. Therefore, at the end of ranking phase $p - 1$ (beginning of ranking phase $p$), every message $m_v, v \in C_{i-p}$, is known to at least $T$ nodes $u \in C_i$. By Property 2 for $p$ (which is already proved) and $i$, all messages $m_v, v \in C_{i-p}$ known in $C_i$ are fresh, which completes the proof. □

Induction step for Property 4. By Property 4 for $p - 1$ and $i$, for every node $u \in C_i$, every node $v$ such that $v \in C_j$ for some $i - p + 1 \leq j \leq i$, it holds that $m_v \in R_u(\bar{t}_{p-1})$, and $m_v$ is non-fresh. This holds also at the end of ranking phase $p$. We still need to show that the property holds for all message $m_v, v \in C_{i-p}$. Notice that Properties 1,2 and 3 are already proved for $p$.

By Property 3 for $p$ and $i$, at the beginning of ranking phase $p$, every message $m_v, v \in C_{i-p}$, is fresh in at least $T$ nodes $u \in C_i$. By Property 2 for $p$ and $i$, at the beginning of ranking phase $p$, every node $u \in C_i$ has at most $8\tau$ fresh messages. By Lemma 4.10 for $p$, all are sent during ranking phase $p$. This means that every message $m_v, v \in C_{i-p}$, is sent by at least $T$ nodes of the clique $C_i$. This implies that every
message \( m_v, v \in C_{i-p} \), is received by every node \( u \in C_i \) at least \( T \) times. Thus, at the end of ranking phase \( p \), every message \( m_v, v \in C_{i-p} \) is known and non-fresh in all nodes \( u \in C_i \), which completes the proof.

Property 4 guarantees that full information spreading is completed after ranking phase \( p = n/k \), with probability at least
\[
1 - \left( \frac{2n/k}{n^{d-2}} + \frac{2}{n^{\alpha/48-\epsilon}} \right) \geq 1 - \left( \frac{1}{n^{d-2}} + \frac{1}{n^{\alpha/48-\epsilon}} \right) \geq 1 - \frac{1}{n^c},
\]
for a constant \( c \), by fixing \( d \) and \( \alpha \) to values \( d > c + 3, \alpha > 48c + 96 \). This completes the proof of Lemma 4.4, from which Theorem 4.3 follows.
Chapter 5

Fault Tolerance

Algorithm 4.1 highly depends on the random phase in the following sense. For every node $v$, consider the set of nodes in neighboring cliques that know message $m_v$ by the end of the random phase. Then, w.h.p. the algorithm spreads $m_v$ using the layers of nodes in the above set (“carriers”). This means that the paths of a message are fixed very early in the algorithm and do not alternate.

A single failure of a node in each layer (carrier) is sufficient to break down its role. Each message relies on at least $T$ different layers to proceed. Hence, the algorithm is sensitive to failures in which less than $T$ carrier layers are non-faulty.

At the beginning of ranking phase $p$, consider the case where a message $m_v \in C_i-\rho$ is fresh in $x < T$ nodes in clique $C_i$, due to failures. The behavior of the algorithm in such case is as follows: During the ranking phase, less than $T$ nodes in the clique send the message, so all other nodes in $C_i$ receive the message less than $T$ times, thus it stays fresh in all of them at the end of ranking phase $p$. Starting from the next ranking phase, the message $m_v$ propagates regularly over those $x < T$ carriers, but also propagates over all other carriers, with a delay of a phase. This means that every layer becomes responsible for one extra message (in addition to at most $8\tau$ messages), which may still be tolerable. In general, our algorithm can manage a constant number of such occurrences.

We aim to cope with a larger number of failures, so we modify our algorithm to help layers bypass their failing nodes, so they continue operating as carriers.

5.1 Shuffle Phases

We invoke a shuffle phase between every two ranking phases, so phases of the algorithm now proceed as described in Figure 5.1.

Roughly speaking, the objective of a shuffle phase, is that nodes of every clique re-divide their responsibilities over messages. A shuffle phase consists of $8\tau$ rounds. During it, every node sends its fresh messages (and receives fresh messages from all neighbors). Instead of updating the regular $cnt$ values, nodes use separate counters, $phasecnt$, to
count the number of receptions for each message during the current shuffle phase. Recall that the objective is shuffling the fresh messages between nodes of same clique. Thus, at the end the of the shuffle phase, every node identifies and filters out unwanted messages, which are messages received from neighboring cliques (low phasecnt values), and messages that where already non-fresh prior to the start of the shuffle phase. Then it randomly picks $4\tau$ new fresh messages, to start the next ranking phase with.

The important gain from this cooperative division of responsibilities done by the nodes of a clique, is that a node $u \in C_i$ that does not receive new messages from its faulty neighbor $u' \in C_{i-1} \cap L(u)$, can overcome the failure of the carrier layer, and still take part in transmitting relevant messages from one clique to the other, with no delays. Figure 5.2 shows an illustration of the new expected routes on which a message originated at $C_1$ is relayed to reach $C_{n/k}$.

**Theorem 5.1.** Algorithm 5.1 completes full information spreading on $G_{n,k}$ in $O\left(\frac{n}{k} \log^3 n\right)$ rounds, w.h.p.

Proving the four properties for the modified algorithm implies Lemma 4.4, from which Theorem 5.1 follows. In the previous analysis, the transition from the end of a ranking phase to the beginning of the next one was immediate, therefore claims that hold at end of ranking phase $p-1$, automatically hold at the beginning of ranking phase.
Algorithm 5.1 for each node $u$

1: `SyncRound$(m_u)$` \hspace{1cm} ▷ Round 0
2: RandomPhase()
3: loop
4: RankingPhase()
5: ShufflePhase()
6: end loop

**ShufflePhase** $p$

7: $\hat{B}_u(t_p) \leftarrow$ fresh messages in $B_u(t)$ \hspace{1cm} ▷ $t = t_p$
8: for all $m_v \in \hat{B}_u(t_p)$ do
9: $\text{phasecnt}_{u,v} \leftarrow 1$
10: end for
11: $R \leftarrow \hat{B}_u(t_p)$
12: loop $8\tau$ times
13: if $\hat{B}_u(t_p) = \emptyset$ then
14: send own message $m_u$
15: else
16: pop and send a fresh message from $\hat{B}_u(t_p)$
17: end if
18: $R' \leftarrow$ receive messages
19: for all $m_v \in R'$ do
20: if $m_v \notin R$ then
21: $\text{phasecnt}_{u,v} \leftarrow 1$
22: else
23: $\text{phasecnt}_{u,v} \leftarrow \text{phasecnt}_{u,v} + 1$
24: end if
25: $R \leftarrow R \cup \{m_v\}$
26: end for
27: $t \leftarrow t + 1$
28: end loop
29: $R \leftarrow R$ after filtering out unwanted messages. \hspace{1cm} ▷ Filter out messages $m_v$ with $\text{phasecnt}_{u,v} < \hat{c} \cdot T$
30: $R_u(t) \leftarrow R_u(t) \cup R$ \hspace{1cm} ▷ Filter out messages that were non-fresh prior to the start of the phase
31: Select $4\tau$ messages from $R$ randomly, rank them from 1 to $4\tau$.

$p$. Here, every two consecutive ranking phases are separated by a shuffle phase, implying that $\tilde{t}_{p-1}$ and $t_p$ are not equal anymore. We need to prove that the relevant claims that hold at the beginning of a shuffle phase (end of a ranking phase) hold also at the end of the shuffle phase (beginning of the next ranking phase). That is, we prove that shuffle phases preserve the required properties. The addition of the shuffle phase does not affect the progress of the algorithm until the end of the first ranking phase. Thus, the base case in the inductive proof of the four properties stays as is. Modifications are needed to the proofs of inductive steps.

Before heading to modify the proof of the induction step, we first prove the following,
Lemma 5.2. Assume Properties 2, 3 and 4 hold at the end of ranking phase $p - 1$. Then, for each node $u \in C_i$, and for each direction of flow, at the end of shuffle phase $p - 1$, there are $k$ remaining messages $m_v, v \in C_{i-p}$ (similarly $C_{i+p}$) in $R$ after filtering out unwanted messages (in line 29).

Proof. Properties 2 and 3 hold at the end of ranking phase $p - 1$, i.e., at the beginning of shuffle phase $p - 1$, for every node $v \in C_{i-p}$, it holds that $m_v$ is fresh in at least $T$ nodes $u \in C_i$, and that every node in $C_i$ has at most $4\tau$ fresh messages per direction. Thus, during the shuffle phase, every message $m_v, v \in C_{i-p}$, is sent (and thus, received) at least $T$ times by nodes of $C_i$, and therefore is not filtered out at the end of the shuffle phase. As already discussed, messages that originate at $m_v, v \in C_{i-p-1}$ are filtered out due to low phasecnt values. By Property 4 for end of ranking phase $p - 1$, for every node $v$ such that $v \in C_j$ for some $i - p + 1 \leq j \leq i$, it holds that $m_v$ non-fresh, so they are filtered out. In total, all messages $m_v, v \in C_{i-p}$, are not filtered out, and only them. The other direction of flow is symmetric. □

Lemma 5.3. Assume Properties 2, 3 and 4 hold at the end of ranking phase $p - 1$. Then, with probability at least $1 - \frac{\ln n}{16\alpha} - \frac{1}{n}$, at the end of shuffle phase $p - 1$, every message that is not filtered out in node $u \in C_i$, is selected to be fresh by at least $T$ nodes in $C_i$.

Proof. Assume Properties 2, 3 and 4 hold at the end of ranking phase $p - 1$. Fix $i, v$. Let $1_{u,v}$, for every $u \in C_i$, be indicator variables that indicate whether node $u$ selects $m_v$ at the end of shuffle phase $p - 1$, or not. By Lemma 5.2, there are at most $2k$ remaining messages in $R$ after filtering out unwanted messages (in line 29). Thus, the probability for each message to be within the $4\tau$ selected messages at the end of the shuffle phase is at least

$$\Pr[1_{u,v} = 1] \geq \frac{4\tau}{2k} = \frac{2\tau}{k}. $$

Let $X_v = \sum_{u \in C_i} 1_{u,v}$, be the number of nodes in $C_i$ that select message $m_v$ at the end of shuffle phase $p - 1$. Then

$$\mu = E(X_v) = E\left(\sum_{u \in C_i} 1_{u,v}\right) = \sum_{u \in C_i} E(1_{u,v}) \geq \sum_{u \in C_i} \frac{2\tau}{k} = k \cdot \frac{2\tau}{k} = 2\tau. $$

The indicator variables are independent, as they refer to decisions of distinct nodes. By
applying a Chernoff bound (Lemma 2.2), we get

\[ \Pr[X_v \leq (1 - \delta)\mu] \leq \exp\left(-\delta^2 \frac{\mu}{2}\right) = \exp\left(-\delta^2 \alpha \log n\right) < \frac{1}{n^{\alpha/2}}. \]

By setting \( \delta = \frac{3}{4} \), we get that a message \( m_v \) is selected fresh in at least \( T \) nodes \( u \in C_i \) with probability at least \( 1 - \frac{1}{n^{\alpha/16}} \). By a union bound, this holds for every node \( v \) with probability at least \( 1 - \frac{1}{n^{\alpha/16}} \). \( \Box \)

To match the modification of the algorithm, we show that the four properties now hold for \( p \) with probability at least \( 1 - \left( \frac{2}{n^{\alpha/48}} + \frac{2(p - 1)}{n^{d/2}} + \frac{p}{n^{9\alpha/16}} + \frac{1}{n^{d/2}} + \frac{1}{n^{d/2}} \right) \). To prove the new induction step, we make similar assumptions as earlier when proving the induction step, i.e., all events described in the four properties for \( p - 1 \), and in Lemma 4.10 for \( p - 1 \) and \( p \), occur. In addition, we assume that events described in Lemma 5.3 for \( p - 1 \), occur. In total, this happens with probability at least

\[ 1 - \left( \frac{2}{n^{\alpha/48}} + \frac{2(p - 1)}{n^{d/2}} + \frac{p}{n^{9\alpha/16}} + \frac{1}{n^{d/2}} + \frac{1}{n^{d/2}} \right) = \]

\[ 1 - \left( \frac{2}{n^{\alpha/48}} + \frac{2p}{n^{d/2}} + \frac{p}{n^{9\alpha/16}} \right). \]

**Extension of induction step for Property 1.** The property holds at the end of ranking phase \( p - 1 \). At the beginning of shuffle phase \( p - 1 \), each pioneer message in a clique is known to exactly one node in the clique. Thus, at the end of the shuffle phase, the phasecnt values for pioneer messages are at most 2 (one reception is from the respective node within the same clique, and the other is from the neighbor from the neighboring clique). In conclusion, all pioneer messages are filtered out, so there are no pioneer messages at the beginning of ranking phase \( p \), which completes the proof. \( \Box \)

**Extension of induction step for Property 2.** The property holds at the end of ranking phase \( p - 1 \). At the beginning of shuffle phase \( p - 1 \), considering one direction of flow, all fresh messages \( m_v \) in nodes of clique \( C_i \) originate at nodes \( C_{i-p} \), except for pioneers (originating at nodes in \( C_{i-p-1} \)). At the end of shuffle phase \( p - 1 \), as already discussed, all pioneer messages are filtered out due to low phasecnt values. By Property 4 for the end of ranking phase \( p - 1 \), all messages \( m_v \notin C_{i-p} \) are non-fresh, so they are filtered out (if any) for being non-fresh prior to the start of shuffle phase \( p - 1 \). Thus, in total, considering both directions, at the end of shuffle phase \( p - 1 \), each node selects \( 4\tau \) of the messages \( m_v, v \in C_{i-p} \cup C_{i+p} \), marks them fresh and ranks them 1 to \( 4\tau \). This completes the proof. \( \Box \)

**Extension of induction step for Property 3.** Properties 2 and 3 hold at the end of ranking phase \( p - 1 \), i.e., at the beginning of shuffle phase \( p - 1 \), for every node \( v \in C_{i-p} \), it
holds that $m_v$ is fresh in at least $T$ nodes $u \in C_i$, and that every node in $C_i$ has at most $4\tau$ fresh messages per direction. During the shuffle phase, every message $m_v, v \in C_{i-p}$, is sent at least $T$ times by nodes of $C_i$, and therefore is not filtered out at the end of the shuffle phase. By Lemma 5.3 for $p-1$, each message is selected and becomes fresh in at least $T$ nodes, which completes the proof.

\[ \square \]

**Extension of induction step for Property 4.** The original proof of Property 4 for $p$ shown in the previous section relies on Property 4 at the end of ranking phase $p-1$, on Properties 2 and 3 at the beginning of ranking phase $p$, and on Lemma 4.10 for $p$. At this point, all of them are proved. Thus, the same original proof for Property 4 applies directly.

In other words, by Property 4 for $p-1$ and $i$, for every node $u \in C_i$, every node $v$ such that $v \in C_j$ for some $i - p + 1 \leq j \leq i$, it holds that $m_v \in R_u(\bar{t}_{p-1})$, and $m_v$ is non-fresh. Notice that shuffle phases preserve this. By Property 3 for $p$ and $i$, at the beginning of ranking phase $p$, every message $m_v, v \in C_{i-p}$, is fresh in at least $T$ nodes $u \in C_i$. By Property 2 for $p$ and $i$, at the beginning of ranking phase $p$, every node $u \in C_i$ has at most $8\tau$ fresh messages. By Lemma 4.10 for $p$, all are sent during ranking phase $p$. This means that every message $m_v, v \in C_{i-p}$, is sent by at least $T$ nodes of the clique $C_i$. This implies that every message $m_v, v \in C_{i-p}$, is received by every node $u \in C_i$ at least $T$ times. Thus, at the end of ranking phase $p$, every message $m_v, v \in C_{i-p}$ is known and non-fresh in all nodes $u \in C_i$, which completes the proof. $\square$

This completes the proof. Recall that we assumed that all events described in the four properties for $p-1$, in Lemma 4.10 for $p-1$ and $p$, and in Lemma 5.3 for $p-1$, occur. Thus, the properties are proved with probability at least $1 - \left( \frac{2}{n^{2d-1}} + \frac{2p}{n^{d-2}} + \frac{p}{n^{9d-16-1}} \right)$.

Assigning $p = n/k$ in Property 4 proves Lemma 4.4, from which Theorem 5.1 follows, with probability at least

\[
1 - \left( \frac{2}{n^{2d-1}} + \frac{2n/k}{n^{d-2}} + \frac{n/k}{n^{9d-16-1}} \right) \geq 1 - \frac{1}{n^{c}},
\]

for a constant $c$, by fixing $d$ and $\alpha$ to values $d > c + 3, \alpha > 48c + 96$.

### 5.2 Resilience to Faults

Recall that we consider a model of independent failures of nodes, where each node fails at each round with probability $q$, and never recovers.

Let $\tau_e \leq 2^{n/\tau} = O \left( \frac{n}{\tau} \log^3 n \right)$ (the round number at the end of ranking phase $n/k$ in Algorithm 5.1). We show that the algorithm tolerates failures for $q \leq 1/(32\tau_e)$. First, we prove the following:
Lemma 5.4. At the end of round $\tau_e$, the number of non-faulty nodes in each clique is at least $\left(\frac{30k}{32}\right)$, with probability at least $1 - \frac{1}{n^{31}}$.

Proof. Let $1_u$, for every node $u$, be an indicator variable that indicates whether node $u$ is non-faulty after $\tau_e$ rounds, or not. Then

$$\Pr[1_u = 1] = (1 - q)^{\tau_e} \geq 1 - q\tau_e \geq 1 - 1/32 = 31/32.$$  

Let $X_i = \sum_{u \in C_i} 1_u$, for every $i, 1 \leq i \leq n/k$, be the number of non-faulty nodes in $C_i$ after $\tau_e$ rounds. Then

$$\mu = E(X_i) = E\left(\sum_{u \in C_i} 1_u\right) = \sum_{u \in C_i} E(1_u) \geq \sum_{u \in C_i} 31/32 = 31k/32.$$  

The indicator variables are independent, as failure events of nodes are independent. By applying a Chernoff bound (Lemma 2.2), with $\delta = \frac{1}{31}$, we get

$$\Pr\left[X_i < \frac{30k}{32}\right] \leq \Pr\left[X_i \leq (1 - \delta)\frac{31k}{32}\right] \leq \Pr[X_i \leq (1 - \delta)\mu] \leq \exp\left(-\delta^2\frac{\mu}{2}\right) \leq \exp\left(-\frac{\delta^231k}{2 \cdot 32}ight) < \frac{1}{n^{31n^{2}}}.$$  

The inequality in second line holds because $k = \Omega(\log^3 n)$. We get that at the end of round $\tau_e$, the number of non-faulty nodes in a clique is at least $\left(\frac{30k}{32}\right)$ with probability at least $1 - \frac{1}{n^{31n^{2}}}$. By a union bound, this holds for every clique with probability at least $1 - \frac{1}{n^{31}}$. $\square$

Theorem 5.5. Algorithm 5.1 completes full information spreading on $G_{n,k}$ in $\tau_e = O\left(\frac{n}{k} \log^3 n\right)$ rounds, for any node failure probability per round $q$, $0 \leq q \leq \frac{1}{32\tau_e}$, w.h.p.

Proof. Fix $i, p$. Let $m_v$ be a message that is fresh in at least $T$ (non-faulty) nodes in $C_{i-1}$ at the end of shuffle phase $p - 1$. Here we analyze the probability that $m_v$ is not shuffled successfully in clique $C_i$. An unsuccessful shuffle might occur either because the phasecnt values in $C_i$ at the end of shuffle phase $p$ are smaller than the threshold of $T^* = \hat{c}T$, so the message is filtered out (denote this event by $A$), or because the message was selected by less than $T$ (non-faulty) nodes. By Lemma 4.10, at the beginning of shuffle phase $p$, the message $m_v$ is supposed to be fresh in at least $T$ nodes in $C_i$ (each of them gets the message from its respective neighbor in $C_{i-1}$). Of these nodes in $C_i$, if one does not send $m_v$ during shuffle phase $p$, then either the node
or its neighbor in \( C_{i-1} \) (or both) becomes faulty by the end of shuffle phase \( p \). The probability, \( \hat{q} \), for such a pair of nodes not to fail is bounded from below (Lemma 2.1) by \( \hat{q} = ((1-q)\tau)^2 \geq (1-q \tau_e)^2 \geq 1-2q \tau_e \geq 1-1/16 \). Fix a set of \( T \) pairs of nodes \( S(m_v) \subseteq C_{i-1} \times C_i \), of those who know message \( m_v \) in \( C_{i-1} \) at the end of shuffle phase \( p-1 \), and their respective neighbors in \( C_i \). There might exist more than \( T \) such pairs, but by fixing a set of size \( T \) and ignoring the rest, we bound the probability of an unsuccessful shuffle from above, as the ignored nodes can only help and increase the probability of success. A “surviving” pair is a pair of nodes from \( S(m_v) \) where both are non-faulty at the end of the shuffle phase, and hence function properly (by sending message \( m_v \)) during shuffle phase \( p \). Denote by \( s \), the number of “surviving” pairs. We have that

\[
\Pr[A] \leq \sum_{s=0}^{T^*-1} \binom{T}{s} \cdot (\hat{q})^s \cdot (1-\hat{q})^{T-s} \leq \sum_{s=0}^{T^*-1} \binom{T}{s} \cdot (1-\hat{q})^{T-s} \\
\leq \sum_{s=0}^{T^*-1} \binom{T}{s} \cdot (1/16)^{T-s} .
\]

We sum over all \( s \in \{0, \ldots, T^*-1\} \), where the number of “survivors” is lower than the threshold of \( cT \), which implies that the message \( m_v \) is filtered out, improperly, at the end of the shuffle phase due to a low \( \phi_{\text{scen}} \) value. By setting \( 0 < \hat{c} \leq \frac{1}{2} \), we get the following,

\[
\Pr[A] \leq T^* \cdot \left( \frac{T}{T/2} \right) \cdot (1/16)^{T/2} \leq cT \cdot \left( \frac{T}{T/2} \right)^{T/2} \cdot (1/16)^{T/2} \leq \\
\leq T/2 \cdot \left( (2e)^{1/2} \right)^T \cdot (1/16)^{T/2} \leq \frac{1}{4} \alpha \log n \cdot \left( (2e)^{1/2} \right)^{1/4} \log n \cdot \left( \frac{1}{2^{1/4}} \alpha \log n \right)^{1/4} \leq \\
\leq n \cdot \left( \frac{2}{\alpha} \right)^{1/4} \log n \cdot \left( \frac{1}{2^{1/4}} \alpha \log n \right)^{1/4} \leq n \cdot n^2 \cdot \left( \frac{1}{n^2} \right) \leq n^2 \alpha^{-1} \cdot \frac{1}{n^\alpha} = \\
= \frac{1}{n^\alpha-\frac{2}{\alpha} n^{-1}} = \frac{1}{n^{\alpha/3-1}} .
\]

Namely, the message is not filtered out with probability at least \( \frac{1}{n^{\alpha/3-1}} \). The number of non-faulty nodes in each clique is at least \( 31k/32 \) with probability at least \( 1 - \frac{1}{n^\alpha} \), by Lemma 5.4. An analysis similar to the one in the proof of Lemma 5.3 (with \( \delta = 11/15 \)) gives that, once the message is not filtered out, it is selected by at least \( T \) of the non-faulty nodes in \( C_i \) with probability at least \( 1 - \frac{1}{n^{\alpha/2}(15/16)} \). In total, by using a union bound, a message is not shuffled successfully between two consecutive shuffle phases with probability at most \( \frac{1}{n^\alpha} + \frac{1}{n^{\alpha}(15/16)} + \frac{1}{n^\alpha} \leq \frac{1}{n^\alpha} \) (for value of \( \alpha \) fixed earlier).

We use union bound two more times, for all messages and for all phases, and get an upper bound for the probability that a message is not propagated properly, of \( \frac{1}{n^\alpha} \). This proves that the algorithm tolerates failures that occur with probability \( 0 \leq q \leq \frac{1}{n^\alpha} \) in the given model, with probability at least \( 1 - \frac{1}{n^\alpha} \). \( \square \)
Chapter 6

Discussion

6.1 Summary

In this research, we show an information spreading algorithm, and prove that it is fast and robust for $G_{n,k}$. By making minor changes to $G_{n,k}$ we can cover additional graphs with same or similar analysis. We believe that the same approach works for additional families of $k$-vertex-connected graphs, such as $k$-steady-growth (discussed below). The intriguing open question is whether this approach can work for general $k$-vertex-connected graphs.

To summarize, we find the question of devising a fast and robust information spreading algorithm in the Vertex-Congest model an intriguing open question, and view our result as a first step in this direction. The technique our algorithm leverages, of using probability distributions that change over time according to how the execution unfolds, may have applications in other settings as well.

6.2 Static-Routes Algorithms

Let $ALG$ be an algorithm that spreads information on $k$-vertex-connected graphs in $O\left(\frac{n}{k} \cdot \text{polylog}(n)\right)$ rounds, by constructing static routes, and using them to disseminate messages in parallel, each message on a specific route. This makes $ALG$ very sensitive to failures, as a single failure in a route suffices to render the entire route faulty.

However, it can easily be configured so that vertex-disjoint routes are combined into groups of size $\gamma$, and every node duplicates its messages and sends them concurrently over these components. Notice that in $k$-vertex-connected graphs, $\gamma$ is bounded from above by $k$. This costs $\gamma$ slowdown in runtime as a trade-off. Denote this configuration of the algorithm by $ALG(\gamma)$. We are interested in cases where $\gamma = O(\text{polylog}(n))$, so that the runtime of the algorithm remains $O\left(\frac{n}{k} \cdot \text{polylog}(n)\right)$. Every combination of $\gamma$ vertex-disjoint routes induces a $\gamma$-vertex-connected subgraph, as it stays connected after the removal of any $\gamma - 1$ vertices. Each component functions as long as it stays connected. According to [CHGK14b, Theorem 1.5], for $\gamma = \Omega(\log^3 n)$, such a component stays...
connected w.h.p. if its nodes are sampled independently with a constant probability. By considering the sampling process imposed by failures, i.e., considering the non-faulty nodes as sampled, then each component stays connected if a constant fraction of its nodes stays non-faulty during the execution, tolerating a constant fraction of nodes that fail. The additional slowdown factor for each message to spread over such a component in the presence of faults can be loosely bounded form above by $O(\gamma)$, as the size of the combined component is $O(\gamma)$ the size of its original routes, (in the worst case a message traverses over all non-faulty nodes of the component). In total, this configuration of the algorithm tolerates the failure of a constant fraction of nodes during its execution, which matches a probability of failure of $q = O\left(\frac{k}{n \cdot \text{polylog}(n)}\right)$ per round, while preserving a time complexity of $O\left(\frac{k}{n} \cdot \text{polylog}(n)\right)$.

The algorithm presented in [CHGK14b, CHGK14a] is static-route, as it constructs CDS packings and routes messages over them. The CDS packings are only fractionally vertex-disjoint, which requires a few modifications to the above analysis. However, despite the above fix, the algorithm remains vulnerable due to the preprocessing stage. The preprocessing stage involves algorithms for connected components identification [Thu95], building a bipartite graph and computing a maximal matching on it (by applying the maximal independent set algorithm [Lub86] on the edges). These algorithms are very sensitive to failures. Some of them are based on communication done over spanning trees, while others incrementally construct their output while assuming that decisions they made earlier still hold. Hence, tolerating failures that occur during the preprocessing stage is more complicated, and the construction of CDS packings in the presence of failures is still an open problem.

### 6.3 Steady Growth Graphs

In the efforts of solving the problem for a larger spectrum of $k$-vertex-connected graphs, we tried to figure out the attributes in graphs $G_{n,k}$ that were essential for the information spreading progress. We give a definition for $k$-steady-growth graphs, a class of $k$-vertex-connected graphs that meet a set of conditions, on which we believe that an algorithm of similar approach to ours performs as good as our algorithm does on $G_{n,k}$. We give an intuitive description of the conditions, and then give a formal definition.

A key requirement is that when a message is first received by a node, it is likely to be either quickly relayed or received many more times within a short time. This keeps the group of new messages in each node of a manageable size. In order for this to happen, we set two conditions. First, we notice that the fact that new messages are received in a stable rate helps in the sense that it gives each node the opportunity to play a part in relaying messages before newer ones approach. Hence, one condition is that for every node $v$, the number of nodes in the $i$-neighborhood of $v$ grows steadily (more specifically, linearly) as $i$ grows. Another important condition on every pair of nodes $v$ and $u$, is that if $i$ is the distance between $v$ and $u$, then there exist many vertex-disjoint
paths connecting them, of lengths roughly $i$. This quickly decreases the number of new messages in a node, as they are likely to be received a sufficient number of times within a short time, allowing the node to realize that the message is not new anymore.

For every node $v$, denote by $D_i(v)$ the set of nodes in distance $i$ from $v$. Notice that for every $k$-vertex-connected graph it holds that $|D_i(v)| \geq k$ for every $v$ and every distance $i > 0$, except for (maybe) the set of farthest nodes from $v$, and the empty sets beyond it (for larger values of $i$). A $k$-vertex-connected graph is called $k$-steady-growth if there exist constants $c_1 \geq 1$ and $0 < c_2 \leq \frac{1}{3}$ such that the following holds for every node $v$: (i) for every distance $i$, $|D_i(v)| \leq c_1 k$, and (ii) for every distance $i \geq 2$, every subset of $D_i(v)$ of size $\geq c_2 k$ has a matching in $D_{i-1}(v)$ of size $\geq c_2 k$.

The following observation might be a good starting point when thinking of an information spreading algorithm that proceeds in phases, that is fast for $k$-steady-growth graphs. For every node $v$, distance $i$, and $u \in D_i(v)$, all neighbors of $u$ are in the union of the three sets $D_{i-1}(v) \cup D_i(v) \cup D_{i+1}(v)$, which means that at least $k/3$ of them are in the same set, forming a subset that is connected by matchings all the way back to node $v$, producing a large number ($\geq c_2 k$) of vertex-disjoint paths between nodes $v$ and $u$.

We believe that an algorithm could be devised to behave as follows, and complete information spreading in such graphs. The algorithm proceeds in phases, such that by the end of phase $i$ it guarantees that the message $m_v$ of every node $v$ reaches every node $u \in D_i(v)$, assisted by its group of $k/3$ neighbors. Phases might consist of smaller units to cover the different cases where the groups are in $D_{i-1}(v)$, in $D_i(v)$, or in $D_{i+1}(v)$.

It remains open whether the above indeed holds, and whether a different algorithm can be constructed for all $k$-vertex-connected graphs.
Bibliography


who graphed several $n/k$-cliques, as a total of $G_{n,k}$
graphed in which each $k$-clique is composed of $n/k$ vertices. The graph $G_{n,k}$ was studied in the context of matching, where the number of graph edges is the total $k$-cliques, each one a graph composed of $n/k$ vertices.

These studies revealed several key findings in the context of matching, where the number of graph edges is the total $k$-cliques, each one a graph composed of $n/k$ vertices.

Vertex-k-vertex connections between vertices are a crucial element in this model. The goal of the study was to develop an algorithm for the dissemination of information on graphs, which would be faster and more efficient than the Congest algorithm. Failures of this type occur during the course of the run of the research.

A major contribution of the research was to set an upper bound on the time of the random version $(\Omega(n/\sqrt{k}))$ of the study. This was achieved by using random algorithms, which define random connections as more frequent.

The algorithm is described in two stages. The first stage describes the basic structure. In the second stage, we identify $G_{n,k}$-supergraphs of the graph $O(n \cdot \log^3(n)/k)$ and prove that at this step, the network is not suitable, and we combine this with a mechanism that improves the algorithm.

In the improved version of the algorithm, the network is stable for the duration of the run of the algorithm, while maintaining the upper bound of the time complexity.

The main contribution of the study is the introduction of a new mechanism for information dissemination on graphs, which is significantly faster and more efficient than existing algorithms.
Network Coding

At the core of this work, we are interested in the problem of broadcasting a message in a network, where the goal is to ensure that all nodes in the network receive the same message, even in the presence of node failures. To achieve this, we use network coding, which allows nodes to combine incoming messages and forward their own data, thus reducing the number of transmissions required.

We consider a network with $n$ nodes and each node can communicate with at most $d$ other nodes. The goal is to broadcast a message of size $m$ from a single source node to all other nodes in the network in the minimum number of transmissions.

We propose two algorithms for this problem: store-and-forward and header-based algorithms.

- **Store-and-forward (SAF)**: In this algorithm, nodes buffer and forward messages as they receive them. This is the most straightforward approach, but it requires a large amount of buffer space.
- **Header-based (HB)**: In this algorithm, nodes forward only headers of messages, which reduces the number of transmissions but requires additional overhead.

We analyze the performance of these algorithms and prove bounds on the number of transmissions required under various conditions. For example, we show that the minimum number of transmissions required for SAF is $O(n)$, while for HB it is $O(n \log n)$.

We also consider the problem of broadcasting in the presence of link failures, where some links may fail during the broadcast. We propose a novel algorithm that can handle link failures by dynamically reconfiguring the network.

Finally, we compare the performance of our algorithms with existing solutions and show that our approaches are more efficient in many cases.
(Vertex-Congest)
המחקר נערך בהנחייתו של פרופסור קרן צנזור-הלל, בפקולטה למדעי המחשב.

תודה

אניatedRoute לאנiatrics העמוקה לפרסמה של, פרופ' קרן צנזור-הלל, על המשרעת במקומיה, ההדרכה והאכפתיות לכל אורך המחקר זה, תודה עמוקה למשותחי הקורא על אสะสมתם וחספנות כל הרגעים. תודה לעמואל זקס ורבעי לעמך על התערבות והמשות.

אני מודה לנטעך על התמיכת הספויות הנדרשות והמשות.
אלגוריתמים להפצת מידעי hakkפת
לتكلחות

نبيור על מחקר

לשם مليון חלוקי השליחה לקבלת התואר
מניחים את הדירוגים המ洉לים המackers

_attack_with WHICH

 görmון כריכ째 במדעי מחשב

טארקט טוקיאן

honış הלסנ טכני - מרכז טכניי לישראל
ארד העש"ה ייפה מרס 2015
אלגוריתמים להפצת מדע שלニュース והסינים לתוכלת

טארק טוקרן