DISSERTATIONES
INFORMATICAE
UNIVERSITATIS
TARTUENSIS
29

IVO KUBJAS

Algebraic Approaches to Problems Arising in Decentralized Systems

DISSERTATIONES INFORMATICAE UNIVERSITATIS TARTUENSIS

29

DISSERTATIONES INFORMATICAE UNIVERSITATIS TARTUENSIS
29

IVO KUBJAS

Algebraic Approaches to Problems Arising in Decentralized Systems

UNIVERSITY OF TARTU
Press
1632

Institute of Computer Science, Faculty of Science and Technology, University of Tartu, Estonia.

Dissertation has been accepted for the commencement of the degree of Doctor of Philosophy (PhD) in computer science on September 3, 2021 by the Council of the Institute of Computer Science, University of Tartu.

Supervisor

Assoc. Prof. Vitaly Skachek
Institute of Computer Science
University of Tartu
Tartu, Estonia

Opponents

Prof. Parastoo Sadeghi
School of Engineering and Information Technology
University of New South Wales, Canberra
Canberra, Australia

Prof. Tadashi Wadayama
Department of Computer Science, Faculty of Engineering
Nagoya Institute of Technology
Nagoya, Japan

The public defense will take place on October 26, 2021 at 10:15 via Zoom.

The publication of this dissertation was financed by the Institute of Computer Science, University of Tartu.

ISSN 2613-5906
ISBN 978-9949-03-707-0 (print)
ISBN 978-9949-03-708-7 (PDF)

University of Tartu Press
http://www.tyk.ee/

To my wonderful wife
To my sweet daughter
To my supporting family
To my excellent teacher
To my great friends

ABSTRACT

With the increased usage of cloud hosting platforms and new wireless technologies, the communication paradigm has changed from server-client models to complex decentralized models. The service providers need to distribute their services across different data centers to be able to handle the enormous traffic loads generated by the customers and to be close to the clients to provide a low level of latency for a good user experience. However, duplicating the data across multiple servers is resource wasteful and cost-inefficient.

In this dissertation, we consider three directions that allow reducing the communication between the servers and users. For all three directions, we represent the problems as mathematical structures and apply algebraic methods to provide solutions to the corresponding problems.

The first problem we consider is data synchronization. In data synchronization, there are nodes with their data sets and their goal is to obtain the union of the sets. A naive approach of exchanging the sets (or even the indexes of the elements) becomes quickly infeasible if the number of data elements grows large. There is a method using invertible Bloom filters (IBFs) which requires transmitting only the number of elements in the symmetric set difference. However, the method requires the knowledge of the number of elements missing in every nodes' set and obtaining this number itself yields significant overhead on the data synchronization protocol. We seek to overcome this limitation in Chapter 3. For that, we present a matrix representation of IBFs and a combinatorial argument to estimate the failure probability of being able to extract only a subset of inserted elements. We call the new representation *partially extractable IBF*. We then present an iterative data synchronization protocol which allows us to obtain the union of the sets even if the size of the symmetric set difference is unknown or inexact. The theoretical results are complemented with experimental simulations.

The second direction we consider is data distribution. In data distribution, the graph representing the network topology can be an arbitrary strongly connected graph. The goal of the nodes is to recover the requested elements from other nodes. In Chapter 4, we formulate a new condition called ρ-solvability of a network topology which gives the minimal number of rounds for any protocol satisfying the nodes' requests. We build on the ideas from index coding and data exchange to propose protocols for any ρ-solvable network. For 1-solvable networks, the described protocol uses a minimal amount of exchanged bits over all possible protocols. For arbitrary ρ-solvable networks, the protocol achieves a minimal number of rounds over all possible protocols.

Finally, we consider the problem of function computation on synchronized data in Chapter 5. This problem differs from data synchronization as the goal of the nodes is to compute the value of a function on the union of the sets. We see that the change in the problem definition allows achieving a significant reduction in the number of transmitted bits. We give an upper bound on the communication

complexity of a family of functions using F-monochromatic rectangles. We also consider a reduction to set intersection and set disjointness functions and obtain corresponding asymptotic bounds. Finally, we describe and study an error-free protocol using a family of hash functions.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE AND ABBREVIATIONS

IBF	invertible Bloom filter
LDPC	low-density parity-check

Set theory

$\mathcal{A}$	set (denoted with calligraphic capital letters)
x	element in set (denoted with lower capital letter)
$\lvert\mathcal{A}\rvert$	number of elements in $\mathcal{A}$
$2^{\mathcal{A}}$	powerset of $\mathcal{A}$
$\mathcal{A}\times\mathcal{B}$	Cartesian product of sets $\mathcal{A}$ and $\mathcal{B}$
$\mathcal{A}^n$	Cartesian product of n sets $\mathcal{A}$
$\mathcal{A}^*$	$\mathcal{A}^* = \cup_{n=0}^{\infty}\mathcal{A}^n$
$\mathcal{A}\triangle\mathcal{B}$	symmetric difference of sets $\mathcal{A}$ and $\mathcal{B}$
$\mathcal{A}\setminus\mathcal{B}$	set difference of sets $\mathcal{A}$ and $\mathcal{B}$
d	size of symmetric difference of sets
$\widetilde{d}$	upper bound on d
$[n]$	set of integers $\{1,\ldots,n\}$
$[m,n]$	set of integers $\{m,\ldots,n\}$
$\{x_i\}_{i\in[n]}$	set with elements x_i, $i\in[n]$

Group theory

$\mathbb{F}$	finite (Galois) field
$\mathbb{F}_p$	finite field of size p
$\mathbb{F}_{p^n}$	finite field of size p^n
$\mathbb{F}^\star$	finite field $\mathbb{F}$ with an additional element $\star$
$\mathbb{N}$	the set of natural numbers

Vector spaces

$\mathbf{x}$	vector (denoted with bold lower case letters)
$\lvert\mathbf{x}\rvert$	length of $\mathbf{x}$
$\mathbf{v}\cdot\mathbf{w}$	inner product of vectors $\mathbf{v}$ and $\mathbf{w}$
$\mathbf{0}$	vector of zeros
$\mathbf{e}_i$	vector with one at position i and zeros elsewhere
$\mathbf{1}$	vector of ones
$(x_i)_{i\in[n]}$	vector with elements x_i, $i\in[n]$
V	vector space (denoted with capital letters)
rowspace$(\mathbf{M})$	vector space induced by rows of $\mathbf{M}$

$W^{\perp}$	orthogonal vector space of W
$U+W$	sum of vector spaces U and W
$U \oplus W$	direct sum of vector spaces U and W

Matrices

$\mathbf{M}$	matrix (denoted with bold upper case letters)
φ	number of matrix rows
κ	number of matrix columns
$\mathbf{I}$	identity matrix
$\mathbf{E}$	all-ones matrix
$\mathbf{Z}$	all-zeros matrix
$\mathrm{diag}(\mathbf{x})$	diagonal matrix induced from vector $\mathbf{x}$
$\mathrm{rank}(\mathbf{M})$	rank of the matrix
$\mathbf{S}_{\varphi\times\kappa}$	set of all $\varphi \times \kappa$ dimensional matrices
$\mathbf{S}^{\mathbb{F}}_{\varphi\times\kappa}$	set of all $\varphi \times \kappa$ dimensional matrices over finite field $\mathbb{F}$
$\mathbf{M}^{[i]}$	i-th row of matrix $\mathbf{M}$
$(\mathbf{M})_{i,j}$	element in i-th row and j-th column of matrix $\mathbf{M}$
$\mathbf{M}^{\top}$	transpose of matrix $\mathbf{M}$
$\mathbf{M}^{n}$	matrix product of $\mathbf{M}$ with itself n times
$\mathbf{M} \otimes \mathbf{N}$	tensor product of matrices $\mathbf{M}$ and $\mathbf{N}$

Complexity theory

$\mathcal{O}(\cdot)$	bounded above asymptotically
$\Theta(\cdot)$	bounded above and below asymptotically
$\Omega(\cdot)$	bounded below asymptotically

Graph theory

$\mathcal{V}$	nodes in graph
$\mathcal{E}$	edges in graph
$\mathcal{G}$	graph $(\mathcal{V},\mathcal{E})$
$\mathbf{D}$	adjacency matrix of a graph
$\mathbf{t}_{s,t}$	path from node s to t
$\mathcal{L}_{s,t}$	all paths from node s to t
$\mu_v(j)$	length of shortest path from node v to any node possessing element x_j
$\mathcal{E}_{\mathtt{out}(v)}$	all outgoing edges from node v
$\mathcal{E}_{\mathtt{in}(v)}$	all incoming edges to node v
$\mathcal{W}_{\mathtt{out}(v)}$	all out-neighbors of v
$\mathcal{W}_{\mathtt{in}(v)}$	all in-neighbors of v

η_v	number of in-neighbors of v
u	number of nodes
σ	number of edges
τ	number of sink nodes

Probability theory

$\Pr_{x \in \mathbb{F}}(I(x))$	probability that indicator function $I(\cdot)$ returns true for uniformly chosen element $x \in \mathbb{F}$
$\Pr(Y = y)$	probability that random value Y obtains value y
$E(Y)$	expected value of random variable Y
$n!$	factorial of n

Protocol notation

Π	protocol
$\text{COMM}(\Pi)$	communication complexity of protocol Π
$\mathcal{P}$	set of all packets
$\mathbf{p}$	transmitted packets
π	number of transmitted packets
π_v	number of transmitted packets by node v
$\mathcal{R}_v$	received packets by node v
r	round count variable
ρ	number of rounds in protocol
$\mathcal{X}$	domain where the nodes sets are sampled from
ω	size of $\mathcal{X}$
n	base-2 logarithm of ω
$\mathbf{x}_v$	has-vector of node v
$\mathcal{S}_v$	has-set of node v
$\mathbf{u}_v$	request-vector of node v
$\mathcal{K}_v$	decoded set of node v
$\mathcal{C}_v$	encoding functions for node v
$\mathcal{D}_v$	decoding functions of node v

Data synchronization

$\mathcal{F}$	IBF
$\mathbf{F}$	IBF represented as state matrix
h	number of hash functions for IBF
β	number of cells in IBF
f	number of elements inserted into IBF
g	number of extracted elements from an IBF
$\mathbf{S}_{\beta,f}$	set of $\beta \times f$-dimensional state-matrices

c	cell in an IBF
r_{Extract}	extraction rate from an IBF
χ	overhead required from an IBF for successful extraction with high probability
$\mathcal{Q}$	Strata Estimator
$\zeta(\beta, f)$	number of stopping matrices in $\mathbf{S}_{\beta,f}$
$\nu(\beta, f, h, g)$	number of state matrices in $\mathbf{S}_{\beta h,f}$ allowing to extract at least g elements

Data dissemination

$\mathbf{P}_v$	set of matrices representing elements possessed by node v
$\mathbf{P}$	set of matrices representing elements possessed by all nodes
$\Gamma(\cdot)$	function returning a representative of $\mathbf{P}_v$
$\Gamma_v(\cdot)$	function returning a representative of $\mathbf{P}_v$ from $\mathbf{P}$
$\mathcal{Z}_v$	indices of elements possessed by node v
$\mathcal{T}_v$	indices of elements requested by node v
$\mathbf{Q}_v$	information matrix of node v
$\mathbf{T}_v$	query matrix of node v
$\mathbf{Y}_v$	transmission matrix of node v
$\mathbf{Y}$	transmission matrix of all nodes
min-rank$(\mathcal{G})$	minimum rank over all matrices which fit $\mathcal{G}$

Function computation

$F(\cdot,\cdot)$	function to be computed
$\Phi(\cdot)$	function to be computed

Hash functions

$H(\cdot)$	hash function
$H_{\text{set}}(\cdot)$	set-hash function
$H_{\text{ch}}(\cdot)$	checksum-hash function
$H_{\text{strata}}(\cdot)$	strata-hash function
$\mathcal{H}$	set of all hash functions
m	length of hash function output
γ	length of checksum-hash function output

Various notation

$x \mid y$	concatenation of bit-string representations of x and y
$\lceil x \rceil$	ceiling function, the least integer greater than or equal to x

1. INTRODUCTION

1.1. Set reconciliation

Set reconciliation problem [49] considers a scenario where two parties A and B possess the sets of data $\mathcal{S}_A$ and $\mathcal{S}_B$, respectively, $\mathcal{S}_A, \mathcal{S}_B \subseteq \mathcal{X}$. The size of the symmetric difference of the two sets d is small when compared to the sizes of $\mathcal{S}_A$ and $\mathcal{S}_B$. The goal of the problem is to design an efficient protocol, such that after it terminates, both parties possess the set $\mathcal{S}_A \cup \mathcal{S}_B$. The number of parties can also be larger than two.

Set reconciliation is useful in practical applications which requires data consistency of across multiple devices [26]. For example, to synchronize data between smart devices in low-power environments or to broadcast routing information [48] in a network.

A naïve protocol, which is based on broadcasting the whole sets by each party, is sub-optimal in the cases where d is small. Several solutions which achieve communication complexity linear in d have been proposed for this scenario. Minsky, Trachtenberg and Zippel were the first to propose a protocol with communication complexity linear in d in [49]. Their approach is based on construction of a characteristic polynomials of the sets and divide the corresponding polynomials. As the common monomials cancel out, then the resulting quotient polynomial contains only the monomials from the elements in the symmetric set difference. The linear communication complexity in d is obtained by representing the polynomials by d evaluation points. The drawback of this approach is that recovering the elements from the quotient polynomial has computational complexity of $\mathcal{O}(d^3)$ due to the need to interpolate a polynomial from d evaluation points.

Goodrich and Mitzenmacher described a data structure called *Invertible Bloom Filters (IBFs)* in [28] and described set reconciliation as one potential application for IBFs. The protocol was described in full by Eppstein, Goodrich, Uyeda and Varghese in [22].

Another approach for optimal set reconciliation was proposed by Skachek and Rabbat in [62] using encoding of sets as vector spaces. Using the encoding, the problem of set reconciliation is reduced to finding a sum of two vector spaces and an efficient algorithm for doing so was given.

The extensions of two-party set reconciliation protocols by using characteristic polynomials and by using IBFs to multi-party setting were considered by Boral and Mitzenmacher in [6] and by Mitzenmacher and Pagh in [50], respectively.

All the previous results on the set reconciliation require existing knowledge of d or an upper bound $\widetilde{d}$ on it. It was shown by Yao in [63] that any deterministic protocol for computing k-element set equality requires k bits of communication. As set equality can be reduced to finding the size of the symmetric set difference, then this also implies that estimating the size of symmetric set difference is expensive in terms of transmitted bits.

A method of random sampling of elements was described by Indyk and Motwani in [30]. With this method, a small subset of the elements from the sets $\mathcal{S}_A$ and $\mathcal{S}_B$ are sampled. The sampled sets can be exchanged to learn the size of the symmetric set difference of the sampled sets. This, in turn can be used to estimate d. However, if $d \ll |\mathcal{S}_A \cup \mathcal{S}_B|$, then for accurate estimation of d, the size of the sampled sets must be relatively large compared to the d.

Another approach to estimate the size of the symmetric set difference is based on using Min-Wise sketches as introduced by Broder in [7] and further studied by Broder, Charikar, Frieze and Mitzenmacher in [8]. Min-Wise sketches use a family of set element permutations to estimate the similarity of two sets, which is defined as

$$\upsilon = \frac{|\mathcal{S}_A \cap \mathcal{S}_B|}{|\mathcal{S}_A \cup \mathcal{S}_B|}.$$

The size of the symmetric set difference d can then be estimated by

$$\frac{1-\upsilon}{1+\upsilon}\left(|\mathcal{S}_A| + |\mathcal{S}_B|\right).$$

Similarly to random sampling, if d is significantly smaller than $|\mathcal{S}_A \cup \mathcal{S}_B|$, then the size of transmitted Min-Wise sketches becomes large.

Strata Estimator protocol uses constant-size IBFs where increasing subsets of the set are inserted into it [22]. When extraction succeeds, the parties can determine the value of d within a factor of two and perform a full reconciliation protocol.

Even as Strata Estimators are more efficient than the other approaches in terms of used transmitted bits, it still requires that the sets $\mathcal{S}_A$ and $\mathcal{S}_B$ are finite. In some applications, it is infeasible to store the complete sets and the algorithms need to operate on data streams [52]. Mayur and Muthukrishnan described updatable sketches in [13] based on Min-Wise sketches to efficiently characterize and compare different streams. The approach of using updatable sketches was developed further by Cormode and Muthukrishnan in [11] and Schweller *et al.* in [60].

Recently, consensus algorithm have been studied with the widespread adaption of blockchain technology. In a consensus, in addition to synchronizing the state of the nodes, they have also to agree on the collective state in the presence of a potentially malicious participant. The study of consensus algorithms was initiated by Lamport in [39] where the Paxos algorithm was proposed. As the Paxos algorithm is complicated, a new consensus algorithm Raft was proposed by Ongaro and Ousterhout in [55].

1.2. Index coding and data exchange problem

Coding on demand by an informed source (*ISCOD*) as a communication problem was introduced by Birk and Kol in [5]. They considered a case where there is a single central server pushing messages to caching clients. The forward channel

from the server to the clients is a fast broadcast channel, but the reverse channel from the clients is slow (this, for example, is a case in satellite communication where clients call in with dial-up).

By using the reverse channel the server learns the contents of the clients caches and their requests. By only requiring that the clients can recover their requested messages, it is possible to significantly reduce number of transmitted bits by the use of algebraic codes. The server constructs an algebraic code on demand by taking into account the clients' caches and their requests. Now, instead of sending all the requested messages, the server only sends the redundant data obtained by encoding the messages.

The problem was further developed by Bar-Yossef, Birk, Jayram and Kol in [2], where they assume that data is given as full error-free messages and every node requests a single data message given by its index (thus, coining the term *index coding*). They showed that the minimum length linear code can be characterized by a function of a graph called *minrank* on the side-information graph of the network and that it is smaller than the bound given by Birk and Kol.

In addition to describing the minimum length of any linear index code, the authors also showed that the matrix achieving minrank of the side-information graph can also be used to define the index code for transmitting the messages. They also showed that this technique is also applicable for almost-linear index codes, where a few transmitted messages can be obtained using non-linear functions.

Even though the previous method gives a precise lower bound, computing the minrank of a side-information graph is a NP-hard problem and thus very slow to compute for arbitrary graphs. Chaudhry and Sprintson presented the reduction to boolean satisfiability (*SAT*) problem [10], which allows to use efficient SAT-solvers for finding a solution for small networks. For larger networks, they proposed several heuristics to simplify the search space for finding an approximate solution.

Network coding was described by Ahlswede, Ning Cai, Li and Yeung as a method to improve transmission rates by employing coding of information from the incoming edges at the intermediate nodes [1]. El Rouayheb, Sprintson and Georghiades showed in [20] that any instance of network coding problem can be reduced to a problem in index coding.

This had two significant implications – first, it can be shown that vector linear coding (where the messages are sub-packetized and linear combinations of the packets are transmitted instead) outperforms simple scalar linear coding (where the encoding functions encode whole messages). Secondly, as shown by Langberg and Sprintson in [40], even finding the universal approximate solution is NP-hard.

The equivalence between network coding and index coding was shown by Effros, El Rouayheb and Langberg in [19], where they additionally gave reductions from network coding to index coding in non-linear case and from index coding to network coding in both linear and non-linear cases.

An alternative view on index coding comes from interference alignment. In

interference alignment problem, there is a potential interference between any two nodes in the network which would allow them to transmit information directly. However, in wireless setting the residual noise may be too large for effectively transmitting information between some node pairs, preventing information transmission in practice. Inversely, in wired setting, the effective channel capacity between some of the node pairs may be infinite (i.e. any amount of information is transmitted instantly). Then, the capacity of the whole network is reduced to the capacity of some transmission bottleneck.

Jafar first showed in [31] that the capacities of the wired and wireless networks are bounded above by the capacity of the index coding problem defined through the corresponding network topology. Furthermore, it was shown that all three problems are equivalent and a solution to the interference alignment problem can be given by a solution to an index coding problem.

This line of research was continued by Maleki, Maleki and Jafar in [47] where the equivalence between interference alignment and index coding was used to show the equivalence between multiple unicast index coding and multiple groupcast index coding, where the optimal code for the latter is known to be non-linear.

Multiple groupcast index coding was introduced by Ji, Tulino, Llorca and Caire in [32] as a generalization of index coding. Compared to (unicast) index coding, in multiple groupgast index coding every node requests more than one message (*multiple*) and there is overlap between the nodes' requests (*groupcast*). By applying techniques from coded caching, they achieve a significant improvement over a naive protocol where every request from the node is considered independently. Furthermore, they are able to prove that the proposed protocol is order-optimal with a constant factor.

Index coding in the presence of errors has been studied from different perspectives. First, Dau, Skachek and Chee studied in [15] the case where the transmitted messages are subject to errors. Kim and No looked in [34] at the alternative case where the side information may have errors but the transmissions are without errors.

Secure index coding is another generalization of index coding where the goal is to transmit messages to receivers in a manner that a receiver which knows some side information can not deduce the transmitted messages. Dau, Skachek and Chee considered the problem in [14] and showed that an index code can be characterized by an error-correcting code. The dimension of the error-correcting code gives the number of total transmissions for the corresponding index code while the minimum distance between the codewords defines the number of side-information messages an eavesdropper may have such that it is not able to recover additional messages. Additionally, the authors developed new constructions for linear index codes which additionally are resistant to random and malicious errors.

The idea of secure index coding was further developed in [43] by Liu, Vellambi, Kim and Sadeghi. They showed that if a legitimate node requests same information what the eavesdropper intends to obtain, it has to know at least two

additional side-information messages. These additional messages can be considered as a secret key between the transmitter and receivers, allowing to achieve information-theoretic security compared to using data encryption.

While most of the works on index coding consider a scenario with a single transmitter, Ong, Ho and Lim initiated the study of multi-sender extension in [54]. They looked at a restricted case where every receiver has only a single message as side-information and every data message is owned only by a single receiver. They allow to have a single or multiple transmitters where the information cached by the transmitters does not have to be complete.

A general form of distributed index coding was studied by Sadeghi, Arbabjolfaei and Kim in [59] where for a side-information set of size k, there can be $2^k - 1$ possible transmitters. They showed that existing lower bounds for single-transmitter index coding instances can be generalized to the distributed case. For the upper bound, they present a grouping strategy of the servers such that the servers in every group collectively encode messages to be transmitted. Liu, Sadeghi, Arbabjolfaei and Kim improved the method in [42] using a method of fractional grouping, where individual transmitters can belong to multiple groups by dedicating a fraction of their capacity.

In index coding problems, the receivers are not able to communicate directly with each other. El Rouayheb, Sprintson and Sadeghi initiated a study of coding for *cooperative data exchange* in [21], where every receiver also becomes a transmitter. They looked at a case where the network topology is a complete graph and the goal of every receiver is to obtain all the missing messages. They presented initial upper and lower bounds on the number of transmissions and gave a deterministic algorithm for encoding the packets. They also showed that the optimal number of transmissions can be presented as a matrix rank minimization problem, which for general case is computationally infeasible to solve.

Courtade and Wesel in [12] and Gonen and Langberg in [27] extended the study to general multihop topologies. Courtade and Wesel represented the problem as a linear optimization problem which can be efficiently approximated. However, they require that the network topology graph is regular, i.e. every node has exactly the same number of neighbours. Gonen and Langberg follow alternative route, reducing the problem to *dominating set* problem. As solving the dominating set problem is NP-hard, then due to reduction to dominating set problem computing the optimal number of transmissions in cooperative data exchange in general topologies is NP-hard. Furthermore, finding the number of transmissions within a fixed constant of exact value is NP-hard.

2. NOTATION AND SYSTEM MODEL

In this chapter, in Section 2.1, we introduce the notation used in the dissertation. Then, in Section 2.2 we describe a graph-theoretic representation of network topologies.

We describe different data exchange models in Section 2.3. We consider three kinds of models - data distribution in Section 2.3.1, data synchronization in Section 2.3.2, and function computation in Section 2.3.3.

We further specify the framework by describing different randomness models in Section 2.4 and by describing protocol notation in Section 2.5. The framework allows to us identify different scenarios, which we list in Section 2.6.

We describe existing results which we use in our dissertation in Sections 2.7 and 2.8.

2.1. Basic definitions

We denote sets with calligraphic symbols, e.g. $\mathcal{S}$ and a powerset of $\mathcal{S}$ as

$$2^{\mathcal{S}} \triangleq \{\mathcal{A} : \mathcal{A} \subseteq \mathcal{S}\}.$$

Cardinality of a set $\mathcal{S}$ is given by $|\mathcal{S}|$. The Cartesian product of sets $\mathcal{A}$ and $\mathcal{B}$ is the set

$$\mathcal{A} \times \mathcal{B} \triangleq \{(a,b) : a \in \mathcal{A}, b \in \mathcal{B}\}.$$

We generalize the Cartesian product of a set $\mathcal{A}$ with itself as

$$\mathcal{A}^n \triangleq \underbrace{\mathcal{A} \times \ldots \times \mathcal{A}}_{n \text{ times}}. \tag{2.1}$$

If (2.1) is for any $n \geq 0$, then we denote it as $\mathcal{A}^*$.

We denote

$$[n] \triangleq \{1, \ldots, n\}$$

and

$$[m,n] \triangleq \{m, \ldots, n\}.$$

We use $\mathbb{F}_p$ to denote integers modulo p, where p is a prime, thus forming a finite field. If p is apparent from context, we also use a short-hand description $\mathbb{F} \triangleq \mathbb{F}_p$. We denote the extension field of $\mathbb{F}_p$ as $\mathbb{F}_{p^n}$.

Vector spaces are denoted with upper-case letters, e.g. V. We use lower-case bold symbols to denote vectors, e.g.

$$\mathbf{x} \triangleq (x_1, \ldots, x_n),$$

where the length of the vector depends on the context. The length of a vector $\mathbf{x}$ is given by $|\mathbf{x}|$. We reserve the notation $\mathbf{0}$ for an all-zeros vector, $\mathbf{1}$ for an all-ones vector and $\mathbf{e}_i$ for a unit vector with 1 at position i and zeros elsewhere.

Let W be a subspace of V. The orthogonal vector space of W is given by

$$W^{\perp} \triangleq \{\mathbf{v} \in V : \forall \mathbf{w} \in W, \mathbf{v} \cdot \mathbf{w} = \mathbf{0}\},$$

where $\mathbf{v} \cdot \mathbf{w}$ denotes the inner product of the two vectors.

Let $U, W \subseteq V$ be two vector subspaces. We define

$$U + W \triangleq \{\mathbf{u} + \mathbf{w} : \mathbf{u} \in U, \mathbf{w} \in W\} \subseteq V.$$

If $U \cap W = \{\mathbf{0}\}$, then we also write $U \oplus W$ instead of $U + W$.

We denote the matrices with bold upper-case letters, e.g. $\mathbf{M}$, where the dimensions of the matrix depends on the context. We use $\mathbf{M}^{[i]}$ to denote the i-th row $\mathbf{M}$ and $(\mathbf{M})_{i,j}$ to denote the entry in the i-th row and j-th column. The transpose of a

matrix $\mathbf{M}$ is given by $\mathbf{M}^\top$. The matrix product of matrices $\mathbf{M}$ and $\mathbf{N}$ is denoted as $\mathbf{MN}$. We denote

$$\mathbf{M}^n \triangleq \underbrace{\mathbf{M}\cdots\mathbf{M}}_{n \text{ times}}.$$

The rank of the matrix $\mathbf{M}$ is the number of linearly independent rows. It is denoted as $\mathrm{rank}(\mathbf{M})$. If the matrix elements are over $\mathbb{F}_2$, then we denote the matrix rank as $\mathrm{rank}_2(\mathbf{M})$.

We denote the set of all $\varphi \times \kappa$-dimensional matrices as $\mathbf{S}_{\varphi\times\kappa}$ and the set of all $\varphi \times \kappa$-dimensional matrices over the finite field $\mathbb{F}$ as $\mathbf{S}^{\mathbb{F}}_{\varphi\times\kappa}$.

We reserve the symbol $\mathbf{I}$ for the identity matrix, i.e the matrix with ones on the diagonal and zeros elsewhere. We reserve the symbol $\mathbf{E}$ for an all-ones matrix and $\mathbf{Z}$ for an all-zeros matrix. For a vector $\mathbf{x} = (x_1, x_2, \cdots, x_n)$, we denote by $\mathrm{diag}(\mathbf{x})$ the $n \times n$-dimensional matrix where the element in i-th row and i-th column is x_i and 0 else.

We use the notation $\mathbf{M} \otimes \mathbf{N}$ for the standard tensor product of the matrices $\mathbf{M}$ and $\mathbf{N}$. If $\mathbf{M}$ is $\varphi \times \kappa$-dimensional and $\mathbf{N}$ is $\varphi' \times \kappa$ dimensional, then the tensor product is $\varphi\varphi' \times \kappa\kappa'$-dimensional matrix

$$\mathbf{M} \otimes \mathbf{N} \triangleq \begin{bmatrix} (\mathbf{M})_{1,1}\mathbf{N} & \cdots & (\mathbf{M})_{1,\kappa}\mathbf{N} \\ \vdots & \ddots & \vdots \\ (\mathbf{M})_{\varphi,1}\mathbf{N} & \cdots & (\mathbf{M})_{\varphi,\kappa}\mathbf{N} \end{bmatrix}$$

We use the notation $\mathrm{rowspace}(\mathbf{M})$ to denote the row space of the matrix $\mathbf{M}$. If $\mathbf{M}$ is $\varphi \times \kappa$-dimensional and the elements of $\mathbf{M}$ are from $\mathbb{F}$, then

$$\mathrm{rowspace}(\mathbf{M}) \triangleq \left\{ \sum_{i\in[\varphi]} \lambda_i \mathbf{M}^{[i]} : \lambda_i \in \mathbb{F} \right\}.$$

2.2. Network model

In what follows, we describe a network topology using graph-theoretic representation. This representation allows us to use the well-known results for describing and studying different network models.

Let the number of nodes in the network be denoted as u. The participating nodes in a decentralized system can be described as a directed graph

$$\mathcal{G} \triangleq (\mathcal{V}, \mathcal{E}),$$

where $\mathcal{V}$ is the set of nodes in the network and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ are the directed edges between the nodes. An edge $e \triangleq (s,t)$ indicates that the source node s can transmit a packet to the sink node t via a direct channel without intermediaries.

For a node $v \in \mathcal{V}$, we define outgoing edges $\mathcal{E}_{\mathtt{out}(v)}$ as a set of edges which have v as a source:

$$\mathcal{E}_{\mathtt{out}(v)} \triangleq \{e : e = (v,t) \in \mathcal{E} \text{ for some } t \in \mathcal{V}\}.$$

For a node $v \in \mathcal{V}$, we define incoming edges $\mathcal{E}_{\mathtt{in}(v)}$ as a set of edges which have v as a sink:

$$\mathcal{E}_{\mathtt{in}(v)} \triangleq \{e : e = (s,v) \in \mathcal{E} \text{ for some } s \in \mathcal{V}\}.$$

The sink nodes in $\mathcal{E}_{\mathtt{out}(v)}$ are called out-neighbors of v and are denoted as

$$\mathcal{W}_{\mathtt{out}(v)} \triangleq \{t : (v,t) \in \mathcal{E}\}.$$

The source nodes in $\mathcal{E}_{\mathtt{in}(v)}$ are called in-neighbors of v and are denoted as

$$\mathcal{W}_{\mathtt{in}(v)} \triangleq \{s : (s,v) \in \mathcal{E}\}.$$

Additionally, we denote the number of in-neighbors of v as η_v.

We assume that the graph contains no self-loops. Self-loops are edges from a node to itself, i.e. (v,v) for every $v \in \mathcal{V}$. Furthermore, we assume that for any two nodes $s,t \in \mathcal{V}$ there is at most one edge $(s,t) \in \mathcal{E}$.

We consider two transmission models – *broadcast* and *unicast* model. In broadcast model, only a single transmission is required to transmit a packet to all out-neighbors of a source node s. In unicast model, a transmission is required to transmit a packet over every edge.

A transmission occurs when the source node s encodes a packet $\mathbf{p}$ and transmits it to sink nodes. We give a more detailed definition of the encoding function and a packet later. We also do not restrict the throughput of the edges.

Example 1 (Unicast and broadcast networks)**.** We illustrate the unicast network in Figure 1 and broadcat network in Figure 2.

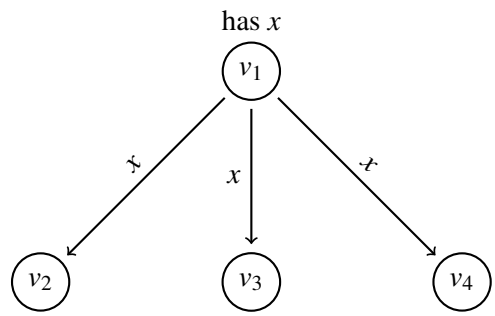

Figure 1. Example of unicast network

In both cases, the underlying graph of the network topology has nodes $\mathcal{V} = \{v_1, v_2, v_3, v_4\}$ with edges $\mathcal{E} = \{(v_1, v_2), (v_1, v_3), (v_1, v_4)\}$. If v_1 transmits element x to all other nodes, then in unicast network this requires three transmissions. In broadcast network, it requires a single transmission. ■

We say that there exists a path from a node s to t if there are edges $e_1, \ldots, e_\sigma$ such that

$$e_1 = (s, v_1)$$

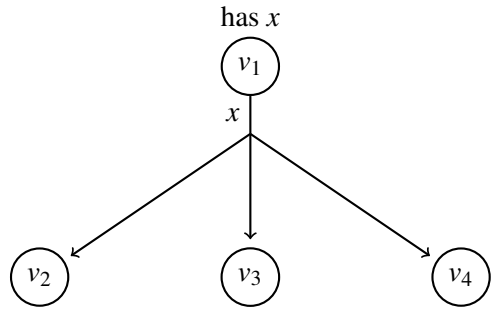

Figure 2. Example of broadcast network

$$\begin{aligned} e_2 &= (v_1, v_2) \\ &\vdots \\ e_\sigma &= (v_{\sigma-1}, t). \end{aligned}$$

We denote the path between s to t as

$$\mathbf{t}_{s,t} \triangleq (e_1, \ldots, e_\sigma)$$

and all possible paths from s to t as $\mathcal{L}_{s,t}$. The length of the path $\mathbf{t}_{s,t}$ is the number of edges in the path and is given by $|\mathbf{t}_{s,t}|$.

The distance from node s to node t is the length of the shortest path between s and t, and infinity if there is no directed path. The diameter of a graph is the maximum distance between any two nodes in the graph.

The graph $\mathcal{G}$ is *strongly connected* if there exists a directed path between any $s,t \in \mathcal{V}$. If the graph is not strongly connected, but there exists an undirected path between every two node in the corresponding undirected graph, then we say that the graph is *weakly connected*.

It is *complete* if for any $s,t \in \mathcal{V}$, $s \neq t$, there exist an edge (s,t). Trivially, any complete $\mathcal{G}$ is also strongly connected.

If the nodes $\mathcal{V}$ in the graph can be partitioned into disjoint sets $\mathcal{W}_1$ and $\mathcal{W}_2$ such that for every edge $e \in \mathcal{E}$, $e = (s,t)$, either $s \in \mathcal{W}_1$ and $t \in \mathcal{W}_2$, or $s \in \mathcal{W}_2$ and $t \in \mathcal{W}_1$, then we say that the graph is *bipartite*.

Furthermore, if $\mathcal{W}_1 = \{s\}$ and every edge $e \in \mathcal{E}$ is $e = (s,t)$ for some $t \in \mathcal{W}_2$, then we say that the graph is a *star*.

Finally, if there are only two nodes in the graph and the graph is complete, then we say that the graph is a *pair*.

Example 2 (Complete graph)**.** A complete graph is illustrated in Figure 3. The nodes are $\mathcal{V} = \{v_1, v_2, v_3, v_4\}$ and there is an edge (s,t) for every $s,t \in \mathcal{V}$, $s \neq t$. ■

Example 3 (Strongly connected graph)**.** A strongly connected graph is illustrated in Figure 4. The nodes are $\mathcal{V} = \{v_1, v_2, v_3, v_4\}$. We see that there are no direct edges between nodes v_2 and v_3, and nodes v_3 and v_4. However, there exist paths $\mathbf{t}_{v_2,v_3} = ((v_2, v_1), (v_1, v_3))$ and $\mathbf{t}_{v_3,v_4} = ((v_3, v_1), (v_1, v_4))$, making the graph strongly connected.

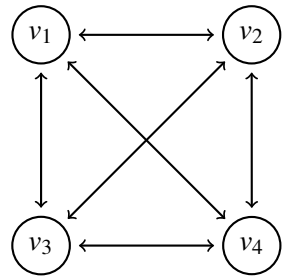

Figure 3. Example of complete graph

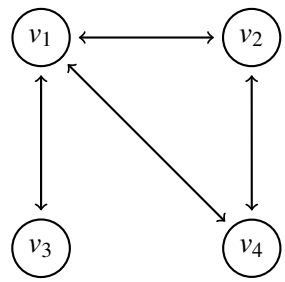

Figure 4. Example of strongly connected graph

■

Example 4 (Bipartite graph)**.** The graph in Figure 5 is bipartite. The set of nodes is $\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\}$. The nodes can be grouped into two distinct sets $\mathcal{W}_1 = \{v_1, v_2\}$ and $\mathcal{W}_2 = \{v_3, v_4, v_5\}$. We see from the illustration that for every edge $e \in \mathcal{E}$, we have $e = (s,t)$ with $s \in \mathcal{W}_1$ and $t \in \mathcal{W}_2$.

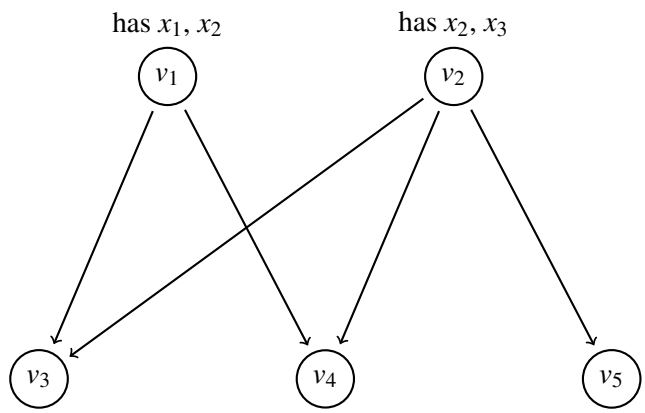

Figure 5. Example of bipartite graph

■

Example 5 (Star graph)**.** A star graph is given in Figure 6. Here, the sink nodes are $\mathcal{W} = \{v_2, v_3, v_4, v_5\}$ and for every $t \in \mathcal{W}$ there exists an edge $(v_1, t) \in \mathcal{E}$.

■

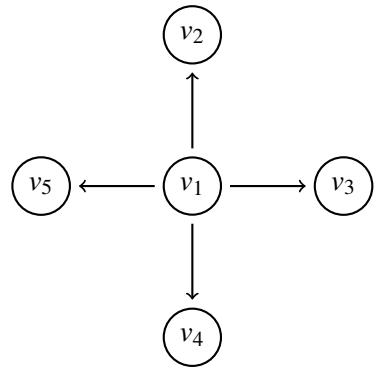

Figure 6. Example of star graph

2.3. Data exchange models

In general, we consider problems which are related to transmitting data between the nodes in a network according to some transmission schedule.

Let $\mathcal{X}$ represent the domain where the data elements are chosen from. We require that the domain is finite. Due to finiteness, every element $x \in \mathcal{X}$ can be identified by its index. Usually, the domain is some finite field $\mathbb{F}$.

We denote the number of elements in the domain $\mathcal{X}$ as

$$\omega \triangleq |\mathcal{X}|$$

and the number of bits to represent any element in $\mathcal{X}$ as n:

$$n \triangleq \lceil \log_2(\omega) \rceil,$$

i.e. the base-2 logarithm of the number of elements in the domain rounded up to the nearest integer.

We associate with the network topology an information vector $\mathbf{x} \in \mathcal{X}^k$ for some $k \geq 0$. Every node $v \in \mathcal{V}$ possesses subvector $\mathbf{x}_v$ of vector $\mathbf{x}$. We call the vector $\mathbf{x}_v$ a *has-vector* of node v. We denote the set which contains all elements in $\mathbf{x}_v$ as $\mathcal{S}_v \subseteq \mathcal{X}$ and call it *has-set* of node v.

We denote the domain of all packets as $\mathcal{P}$.

The transmission of information between nodes happens in rounds. A round consists of the following phases:

1. *computation phase* – nodes apply some encoding functions on their has-vector to obtain a set of packets for every outgoing edge.
2. *transmission phase* – nodes transmit the packets over every outgoing edge. Receiving nodes store the packets for the next phase.
3. *recovery phase* – every node applies decoding functions on their has-vector and received packets to recover additional elements. Recovered elements are added to nodes' has-vectors.

We emphasize that during a transmission phase, the nodes can not use already received packets to update their has-vectors and packets not yet transmitted. This ensures that the rounds can be considered as asynchronous and we do not define the exact order for transmissions within rounds.

After the recovery phase, the current round concludes and new round starts if there are additional packets to transmit. Otherwise, the protocol stops. We denote the total number of rounds in the protocol as ρ and the individual round as r.

We consider two settings – in a *possession oracle* model there is an oracle which knows the indices of elements in every has-vector $\mathbf{x}_v$, $v \in \mathcal{V}$. In the alternative model, there is no such oracle. This distinction leads to corresponding problem statements we consider in this dissertation – data distribution and data synchronization problem.

For the protocols described in this thesis, every node has a particular goal. For example, to obtain additional elements or to compute some value. Thus, we require that the protocols are complete, i.e. the nodes have achieved their goal after protocol execution.

We further describe the notation used to describe a protocol in Section 2.5.

2.3.1. Data distribution problem

In data distribution problem, every node $v \in \mathcal{V}$, in addition to the has-vector $\mathbf{x}_v$, maintains a subvector of unknown elements $\mathbf{u}_v$ from the information vector $\mathbf{x}$. We call the vector $\mathbf{u}_v$ a *request-vector* of node v. The goal of every node in data distribution problem is to recover all elements in their request-vector.

To achieve this goal, the possession oracle takes into account the has- and request-vectors of every node. For every node, it then defines encoding functions for every outgoing edge from that node

$$\mathcal{C}_v \triangleq \{E_{v,e} : e \in \mathcal{E}_{\mathtt{out}(v)}\},$$

where $E_{v,e} : \mathcal{X}^* \to \mathcal{P}^*$ takes as an input the current has-vector of the node and outputs a vector of packets to transmit using the outgoing edge.

The oracle also defines decoding functions for every element in the request-vector

$$\mathcal{D}_v \triangleq \{D_{v,x} : x \in \mathbf{u}_v\},$$

where $D_{v,x} : \mathcal{X}^* \times \mathcal{P}^* \to \mathcal{X}$ takes as an input the current has-vector of the node and received packets from all incoming edges, returning an element in the request-vector.

The encoding and decoding functions $\mathcal{C}_v$ and $\mathcal{D}_v$ are transmitted to the corresponding node v. In the computation phase, every node v computes the packets $\mathbf{p}_{v,e}$ to be sent over every outgoing edge as

$$\mathbf{p}_{v,e} \triangleq E_{v,e}(\mathbf{x}_v).$$

Then, during the transmission phase, every node v uses edge $e = (v,t) \in \mathcal{E}_{\mathtt{out}(v)}$ to transmit every packet in $\mathbf{p}_{v,e}$ to the sink node t. The sink node t stores the received packet in its received packet set $\mathcal{R}_t$.

In the recovery phase, every node v has received and stored a set of packets $\mathcal{R}_v$ from the incoming edges. The nodes apply the decoding functions on their has-vector and received packet set to obtain the decoded element set

$$\mathcal{K}_v \triangleq \{D_{v,x}(\mathbf{x}_v, \mathcal{R}_v) : D_{v,x} \in \mathcal{D}_v\}.$$

The decoded elements are then included in the has-vector of the node.

To generalize the data distribution problem over many rounds, we introduce additional notation. First, we denote the initial has-vector of the node before the protocol starts as $\mathbf{x}_v^{(0)} = \mathbf{x}_v$. The request-vector $\mathbf{u}_v$ is constant over all rounds.

In round r, the encoding and decoding functions are denoted as

$$\mathcal{C}_v^{(r)} \triangleq \{E_{v,e}^{(r)} : e \in \mathcal{E}_{\mathtt{out}(v)}\}$$

and

$$\mathcal{D}_v^{(r)} \triangleq \{D_{v,x}^{(r)} : x \in \mathbf{u}_v\}.$$

However, here the decoding function $D_{v,x}^{(r)}$ may also output $\perp$ which indicates that the element x can not be recovered in round r.

We denote the packets to transmit as

$$\mathbf{p}_{v,e}^{(r)} \triangleq E_{v,e}(\mathbf{x}_v^{(r-1)}).$$

The received packets are denoted as $\mathcal{R}_v^{(r)}$ and the elements are decoded as

$$\mathcal{K}_v^{(r)} \triangleq \{D_v^{(r)}(\mathbf{x}_v^{(r-1)}, \mathcal{R}_v^{(r)}) : D_v^{(r)} \in \mathcal{D}_v^{(r)}\}.$$

The decoded packets $\mathcal{K}_v^{(r)}$ are then used to update the has-vector $\mathbf{x}_v^{(r-1)}$ to obtain $\mathbf{x}_v^{(r)}$. The protocol concludes when $\mathbf{u}_v = \mathbf{x}_v^{(r)}$ for every node v.

2.3.2. Data synchronization problem

In data synchronization problem, the goal of the nodes is to obtain the missing elements from the information vector. In this and the next section the order of the elements does not matter, thus, for every $v \in \mathcal{V}$, instead of has-vector $\mathbf{x}_v$ we consider the corresponding has-set $\mathcal{S}_v$. We denote the union of all has-sets as

$$\mathcal{S}_{\mathtt{REC}} \triangleq \bigcup_{v \in \mathcal{V}} \mathcal{S}_v. \tag{2.2}$$

Every node is missing and seeks to recover elements from the set $\mathcal{S}_{\mathtt{REC}} \setminus \mathcal{S}_v$. However, as in the current setting there is no possession oracle, all nodes share the same encoding function E and decoding function D.

During the computation phase, the nodes apply the encoding function on their has-set to obtain the packets to be transmitted

$$\mathbf{p}_v \triangleq E(\mathcal{S}_v).$$

In transmission phase, the packets are then transmitted to all sink nodes over all edges which store them in their received packet sets $\mathcal{R}_v$.

Finally, in the recovery phase, the nodes apply decoding function on the received packets and their has-set to obtain decoded elements:

$$\mathcal{K}_v \triangleq D_v(\mathcal{S}_v, \mathcal{R}_v).$$

The decoded elements are then added to the nodes' has-sets. As the nodes do not have specific request-sets, then the protocol terminates if the decoded element sets $\mathcal{K}_v$ are empty for all nodes.

To generalize the data synchronization problem over many rounds, we also introduce notation for round-specific sets. We denote the initial has-set of node v as $\mathcal{S}_v^{(0)}$, encoded packets in round r as

$$\mathbf{p}_v^{(r)} \triangleq E(\mathcal{S}_v^{(r-1)}),$$

received packets as $\mathcal{R}_v^{(r)}$, decoded packets as

$$\mathcal{K}_v^{(r)} \triangleq D_v(\mathcal{S}_v^{(r-1)}, \mathcal{R}_v^{(r)})$$

and the has-set is updated as

$$\mathcal{S}_v^{(r)} \triangleq \mathcal{S}_v^{(r-1)} \cup \mathcal{K}_v^{(r)}.$$

The protocol concludes when $\mathcal{S}_v^{(r)} = \mathcal{S}_{\texttt{REC}}$ for all $v \in \mathcal{V}$.

2.3.3. Function computation

In the function computation problem, the goal of the nodes is to cooperatively compute a value of some function F of their has-sets $\mathcal{S}_v$, $v \in \mathcal{V}$. Let the function be $F : (2^{\mathcal{X}})^u \to \mathcal{Y}$, where $u = |\mathcal{V}|$ and $\mathcal{Y}$ is the function range. The goal of nodes is to obtain

$$w \triangleq F(\mathcal{S}_{v_1}, \ldots, \mathcal{S}_{v_u}).$$

Function computation problem can also be considered as a generalization of the data synchronization problem. However, the problem differs from data synchronization as the nodes may not necessarily learn the union of sets $\mathcal{S}_{\texttt{REC}}$ as given in (2.2).

Function computation protocols usually run in rounds. In order to compute the packets to be transmitted in round $r \in [\rho]$, the nodes take into account their

has-sets and all previously received packets. We denote the set of all previously received packets by node v as $\mathcal{R}_v^{(r)}$. We also observe that initially $\mathcal{R}_v^{(0)} = \varnothing$.

Then, in order to compute the packets to be transmitted in round r, the node takes their has-set $\mathcal{S}_v$ and previously received packets $\mathcal{R}_v^{(r)}$ and applies encoding function E:

$$\mathbf{p}_v^{(r)} \triangleq E(\mathcal{S}_v, \mathcal{R}_v^{(r)}).$$

The transmission concludes when all nodes have indicated that they do not have any additional packets to transmit by sending an halting message. Then, the nodes can recover the value w of the function by applying the decoding function on their has-sets and all received packets:

$$w = D(\mathcal{S}_v, \mathcal{R}_v^{(\rho)}).$$

2.4. Randomness models in protocols

We consider different models of how the protocols use randomness. The protocol can either be *deterministic* or *randomized*. In deterministic protocols, the encoding function E does not use any randomness and is uniquely determined only by the has-set $\mathcal{S}_v$ and previously received packets. Thus, for fixed has-sets of the nodes, the transmitted packets are always the same.

In randomized protocols, the encoding function E has access to infinite sequence of unbiased random bits. We follow the discussion in [29] and consider the following special cases of randomized protocol models. In protocols with *shared randomness*, the nodes have access to the same sequence of random bits. Alternatively, in *private randomness* models, all nodes have access to their own sequences of random bits. It is shown in [38] that the number of transmitted packets is in general lower for protocols utilizing shared randomness. Additionally, it is also shown that any private randomness protocol can be simulated by a shared randomness protocol with communication overhead $\mathcal{O}(\log(n))$ for inputs of size n.

Furthermore, we have to consider the failure rate of the randomized protocols. If the protocol occasionally fails to output the correct value, i.e. the failure rate is close to 1 (but not exactly 1), then we call the protocol *Monte-Carlo* style. If the protocol does not fail, then we call such protocol *Las-Vegas* style. In Las-Vegas style protocols, the number of transmitted packets is a random variable and we instead measure the expected number of transmitted bits contrary to Monte-Carlo style protocols, where the number of transmitted bits is exact.

2.5. Notation for describing a protocol

To describe a protocol for data distribution, data synchronization or function computation, we use the notation as given in Protocol 1.

Protocol 1 Example protocol description

Label

Π_1

Network topology

Nodes $\mathcal{V} = \{A, B\}$, edges $\mathcal{E} = \{(A,B),(B,A)\}$.

Randomness model

Private randomness

Input

Information vector $\mathbf{x} = (x_1, \ldots, x_k)$. Node $v \in \mathcal{V}$ possesses subset described by indices $\mathcal{Z}_v$.

Goal

Node $v \in \mathcal{V}$ requests subset of information vector described by indices $\mathcal{T}_v$.

Offline phase

Oracle distributes encoding function E_v to every $v \in \mathcal{V}$.

Online phase

For $r \in [\rho]$

Computation phase

1. Every node $v \in \mathcal{V}$ computes $\mathbf{p}_v^{(r)} = E_v(\mathcal{Z}_v^{(r-1)})$.

Transmission phase

2. Every node v_i transmits $\mathbf{p}_v^{(r)}$ to other node.

Recovery phase

3. Every node $v \in \mathcal{V}$ computes $\mathcal{K}_v = D_v(\mathcal{Z}_v, \mathbf{p}'_v)$.
4. Every node updates has-set as $\mathcal{S}^{(r)} = \mathcal{S}^{(r-1)} \cup \mathcal{K}_v$.

In this notation, the problem instance is described by the following fields: *label* field defines the variable used to denote the protocol for further reference, *network topology* field describes the underlying graph of the network, *randomness model* field defines the used randomness model as in Section 2.4, *input* field defines the initial has-sets of the nodes, and *goal* field defines the requested values of the nodes. The protocol run is given by the following parts: *offline phase* describes the actions of an oracle which ensures that the nodes start the protocol with some specific state, and *online phase* describes the communication as given in Section 2.3.

The communication complexity of the protocol is the number of bits sent during the transmission phases. We denote the communication complexity of the protocol Π as $\mathrm{COMM}(\Pi)$.

2.6. List of problems in decentralized systems for exchanging data

2.6.1. Criteria for data exchange scenarios

By having described different types of network topologies, communication methods, and data exchange models, we can systematically describe different scenarios for exchanging data in decentralized systems.

Graph of the network topology: In networks with many participants, the graph representing the network topology can be a pair, complete, star or connected.

As complete graphs allow to apply 2-party protocols for all possible node pairs, we consider only the scenarios where the communication complexity is decreased due to sending and receiving packets to and from multiple nodes.

Star graphs correspond to the networks where there is a single transmitter and many receivers. This scenario depicts the case of a wireless or satellite station broadcasting data to many receivers in the range.

Finally, we consider all other network topologies which do not fall into the previous types of graphs. To ensure that there is a solution to the problem, we require that the graph is strongly connected.

Transmission model: For networks with two nodes, there is no distinction between unicast and broadcast transmission models. Thus, we only consider unicast transmission model.

As previously mentioned, for complete graphs we only consider the case where we achieve reduction in communication complexity due to coding and thus only consider the broadcast model.

As there is only a single transmitter in a star graph, then we are interested only in the broadcast model.

For strongly connected graphs, we consider both the unicast and broadcast transmission models.

Data exchange model: For networks with two nodes, we do not consider the data distribution model, as due to existence of the possession oracle, the solution is trivial (i.e. the possession oracle transmits the functions which output only the elements in the request-set as packets). For all other graphs with more than two nodes, we consider all possible data exchange models, i.e. data distribution and data synchronization problem.

Function computation: We also look at scenarios which allow to compute the value of a function. As mentioned in Section 2.3.3, the function to be computed takes as inputs all has-sets $\mathcal{S}_v$, $v \in \mathcal{V}$, and as such is only relevant in the context of data synchronization problem. Thus, for function computation, we consider the same scenarios as for data synchronization.

2.6.2. Overview of different data exchange scenarios

Considering the restrictions in Section 2.6.1, we have compiled a list of different scenarios. The scenarios are given in Table 1.

Table 1. Overview of data exchange scenarios

scenario	graph type	trans. model	data exchange
1	pair	unicast	synchronization
2	complete	broadcast	distribution
3	complete	broadcast	synchronization
4	star	broadcast	distribution
5	star	broadcast	synchronization
6	strongly connected	unicast	distribution
7	strongly connected	unicast	synchronization
8	strongly connected	broadcast	distribution
9	strongly connected	broadcast	synchronization
10	pair	unicast	function comp.
11	complete	broadcast	function comp.
12	star	broadcast	function comp.
13	strongly connected	unicast	function comp.
14	strongly connected	broadcast	function comp.

2.7. Invertible Bloom filters

We describe invertible Bloom filters (IBF) [28] for representing a set of elements efficiently. IBF is a probabilistic data structure which allows testing for inclusion of an element in the data structure and listing all stored elements. The inclusion testing using IBF may cause a small rate of false positives and element extraction failure in exchange for decreasing the size of IBF significantly.

In the following, we assume that the domain is finite field $\mathbb{F}$. Usually, the actual used field is a binary extension field $\mathbb{F} = \mathbb{F}_{2^n}$, and the elements in the field are considered to be binary vectors of length n.

2.7.1. IBF construction

IBFs are constructed using hash functions which map inputs from the domain $\mathbb{F}$ to a significantly smaller set $[\beta]$:

$$H : \mathbb{F} \to [\beta].$$

We denote the set of all possible such hash functions as $\mathcal{H}$. We assume that the hash functions can be described efficiently and that they can be indexed, i.e. we can uniquely define $H_i \in \mathcal{H}$ by i.

For the analysis, we require that the hash function output is distributed uniformly. That is, for any fixed $j \in [\beta]$, then for uniformly chosen $x \in \mathbb{F}$ the probability that event $H(x) = j$ happens is exactly $\frac{1}{\beta}$:

$$\forall j \in [\beta] : \Pr_{x \in \mathbb{F}}(H(x) = j) = \frac{1}{\beta}.$$

When constructing an IBF, we choose h hash functions $H_1, \ldots, H_h \in \mathcal{H}$. The hash functions should be pairwise non-colliding for any $x \in \mathbb{F}$, i.e.

$$H_i(x) \neq H_j(x), \text{ for any } i \neq j. \tag{2.3}$$

This can be obtained by splitting $[\beta]$ into h disjoint partitions

$$[1, \beta_1], [\beta_1 + 1, \beta_2], \ldots, [\beta_{h-1} + 1, \beta]$$

and having every hash function H_i output values in the i-th partition.

Additionally, we require another hash function H_{ch} for checksum, which maps elements from domain $\mathbb{F}$ to a smaller field $\mathbb{F}'$:

$$H_{\text{ch}} : \mathbb{F} \to \mathbb{F}'.$$

The invertible Bloom filter is an array of cells `c` with fields `count`, `val` and `ch`. The field `count` contains an integer, field `val` contains an element in $\mathbb{F}$ and the field `ch` contains an element in $\mathbb{F}'$. If we denote the i-th cell as $\mathtt{c}_i$, then invertible Bloom filter is defined as $\mathcal{F} \leftarrow \langle \mathtt{c}_1, \ldots \mathtt{c}_\beta \rangle$.

The procedure Init initializes an IBF $\mathcal{F}$ using given parameters and is described in Algorithm 1. During the procedure, every cell $\mathtt{c}_j$, $j \in [\beta]$, is initialized by setting the value of every field in the cell as identity.

Algorithm 1 Initialize an IBF

```
1: procedure Init(h, β, 𝔽, 𝔽′)
2:     for all j ∈ [β] do
3:         c_j.count ← 0
4:         c_j.val ← 0
5:         c_j.ch ← 0
6:     ℱ ← ⟨c_1, ..., c_β⟩
7:     return ℱ
```

For an initialized IBF $\mathcal{F}$ and an element $x \in \mathbb{F}$, the procedures Insert() inserts the element into IBF, Test() queries if the element is inserted to the IBF and Remove() removes the element if it is already inserted. Finally, the procedure Extract() lists all inserted elements in the IBF.

The procedure Insert() is described in Algorithm 2. It works by iterating over all hash functions H_i, $i \in [h]$, hashing the element x with H_i to obtain an index j_i.

Algorithm 2 Insertion of an element into an IBF

```
1: procedure Insert(F, x)
2:     for all i ∈ [h] do
3:         j_i ← H_i(x)
4:         c_{j_i} ← F[j_i]
5:         c_{j_i}.count ← c_{j_i}.count + 1
6:         c_{j_i}.val ← c_{j_i}.val + x
7:         c_{j_i}.ch ← c_{j_i}.ch + H_ch(x)
8:         F[j_i] ← c_{j_i}
```

Algorithm 3 Test if an element is inserted into the IBF

```
1: procedure Test(F, x)
2:     for all i ∈ [h] do
3:         j_i ← H_i(x)
4:         c_{j_i} ← F[j_i]
5:         if c_{j_i}.count = 0 then
6:             return false
7:     return true
```

This index defines the cell $\mathtt{c}_{j_i}$ to be updated by incrementing the field `count` by one, adding x to field `val` and adding $H_{\text{ch}}(x)$ to field `ch`.

The procedure Test() is described in Algorithm 3. It iterates over the hash functions H_i, $i \in [h]$, to obtain the corresponding indices $\{j_i\}_{i\in[h]}$ and tests if all cells $\mathtt{c}_{j_i}$, $i \in [h]$, contain an element. If so, it returns a positive answer, otherwise rejects.

The procedure can return false-positive for a tested element if the hash functions H_i, $i \in [h]$, are assigned such that every checked cell in $\mathcal{F}$ contains another element.

Algorithm 4 Remove an element from the IBF

```
1: procedure Remove(F, x)
2:     for all i ∈ [h] do
3:         j_i ← H_i(x)
4:         c_{j_i} ← F[j_i]
5:         c_{j_i}.count ← c_{j_i}.count − 1
6:         c_{j_i}.val ← c_{j_i}.val − x
7:         c_{j_i}.ch ← c_{j_i}.ch − H_ch(x)
8:         F[j_i] ← c_{j_i}
```

The procedure Remove() is described in Algorithm 4. It iterates over all hash functions H_i, $i \in [\beta]$, to obtain indices $\{j_i\}_{i\in[h]}$ of the cells to remove the element from. The element is removed from the cell $\mathtt{c}_{j_i}$ by decreasing the field `count` by one, subtracting element x from the field `val` and subtracting $H_{\text{ch}}(x)$ from the field `ch`.

The Remove() procedure should only be called with an element x which has

been inserted to the IBF. If the procedure Remove() is called on an element x which has not been inserted into an IBF, then the cells are updated incorrectly. This could lead to collisions when removing further elements.

2.7.2. Listing elements in IBF

Algorithm 5 Extract all elements inserted into the IBF

```
1:  procedure Extract(F)
2:      S_F ← ∅
3:      repeat
4:          removed ← false
5:          for all j ∈ [β] do
6:              c_j ← F[j]
7:              if c_j.count ≠ 0 then
8:                  (x, ch) ← ( c_j.val / c_j.count , c_j.ch / c_j.count )
9:                  if H_ch(x) = ch then
10:                     S_F ← S_F ∪ {x}
11:                     for all [c_j.count] do
12:                         Remove(F, x)
13:     until removed = false
14:     return S_F
```

The final procedure Extract() differentiates IBF from a standard Bloom filter. With storage overhead, it allows to recover the elements inserted to IBF. The procedure is described in Algorithm 5 and works by iterating over all cells $\mathtt{c}_j$, $j \in [\beta]$ (lines 5-12) to check if the cell can be used to extract an element.

For being able to extract an element from cell $\mathtt{c}_j$ for some $j \in [\beta]$, the following two conditions have to be met:

1. The `count` field has to be positive (line 7).
2. The checksum-hash of the field `val` must be equal to the one stored in the field `ch` (line 9).

In the procedure description, the multiplicity of the element insertion is taken into account when checking for the condition 2, allowing to extract elements which have been inserted multiple times.

If all of the conditions hold, every copy of the element x obtained from the j-th cell is removed from the IBF $\mathcal{F}$ and added to the extracted set $\mathcal{S}_\mathcal{F}$. The inner loop in lines 5-12 continues.

If any elements were extracted during the inner loop, the procedure re-runs it as removing an element from the IBF may have updated a cell in a way which allows for extracting another element. Only when no elements are extracted during a full inner loop, the procedure returns with the set $\mathcal{S}_\mathcal{F}$.

We remark that the procedure modifies the underlying filter. To keep the initial IBF unmodified, a copy of it should be provided to the Extract() algorithm.

We consider

$$\sum_{j=1}^{[\beta]} \mathcal{F}[j].\texttt{count}$$

as the count of all elements inserted in the IBF. As during the Insert() procedure the element is added into h number of cells, then, correspondingly, the count increases by h. Similarly, during removal of an element from the IBF, the count decreases by h. This property allows to count the total number of elements currently contained in the IBF. We denote the cardinality of an IBF $\mathcal{F}$ as f and calculate it as

$$f \triangleq |\mathcal{F}| = \frac{\sum_{j=1}^{[\beta]} \mathcal{F}[j].\texttt{count}}{h}.$$

If after running the procedure Extract() the IBF $\mathcal{F}$ is empty (i.e. $|\mathcal{F}| = 0$), then the procedure succeeded perfectly. Otherwise, the procedure failed, but some elements are still extracted. We can measure the rate of elements extracted by considering the cardinality f of $\mathcal{F}$ before extracting and the number of extracted elements $|\mathcal{S}_{\mathcal{F}}|$. The extraction rate is defined as

$$r_{\mathsf{Extract}} \triangleq \frac{|\mathcal{S}_{\mathcal{F}}|}{f}. \tag{2.4}$$

In the description of the procedure Extract() we can see that if a single element x is inserted multiple times into the IBF, then all copies are removed from the IBF (lines 11 - 12). As this does not change the number of elements extracted from the IBF $|\mathcal{S}_{\mathcal{F}}|$, but does change the cardinality f, then in order to achieve comparable extraction failure probabilities for a fixed extraction rate r_{Extract}, we need to assume that the elements in a particular IBF are unique. The following Lemma 1 shows that for a particular IBF $\mathcal{F}$ with also non-unique elements, there exists an IBF $\mathcal{F}'$ where all elements are unique and the extracted elements $\mathcal{S}_{\mathcal{F}'}$ are equal to $\mathcal{S}_{\mathcal{F}}$.

Lemma 1. *Given an IBF $\mathcal{F}$, the extracted elements set $\mathcal{S}_{\mathcal{F}} = \mathsf{Extract}(\mathcal{F})$ does not depend on the multiplicities of the elements inserted in $\mathcal{F}$.*

Proof. Consider the IBF $\mathcal{F}$ with non-unique elements and the corresponding IBF $\mathcal{F}'$ is initialized using the same hash functions containing the same elements only once. Let

$$\mathcal{S}_{\mathcal{F}} \triangleq \mathsf{Extract}(\mathcal{F})$$

and

$$\mathcal{S}_{\mathcal{F}'} \triangleq \mathsf{Extract}(\mathcal{F}').$$

If $\mathcal{S}_{\mathcal{F}} \neq \mathcal{S}_{\mathcal{F}'}$ then there must exist $x \in \mathcal{S}_{\mathcal{F}} \setminus \mathcal{S}_{\mathcal{F}'}$ or $x \in \mathcal{S}_{\mathcal{F}'} \setminus \mathcal{S}_{\mathcal{F}}$. This implies that there must exist multiplicity $\texttt{count} > 1$ such that

$$\frac{\texttt{count} \cdot x}{\texttt{count}} \neq x$$

or

$$\frac{\texttt{count} \cdot H_{\text{ch}}(x)}{\texttt{count}} \neq H_{\text{ch}}(x),$$

which is a contradiction. Thus $\mathcal{S}_{\mathcal{F}} = \mathcal{S}_{\mathcal{F}'}$. □

In [28], the probability of procedure $\mathsf{Extract}()$ succeeding was only considered for the case where $r_{\mathsf{Extract}} = 1$. The result is given as Theorem 2.

Theorem 2 ([28, Theorem 1])**.** *Define χ_h as*

$$\chi_h^{-1} \triangleq \sup\{\alpha : 0 < \alpha < 1; \forall x \in (0,1), 1 - e^{-h\alpha x^{h-1}} < x\}.$$

Then, as long as β is chosen such that

$$\beta > (\chi_h + \varepsilon) f_0$$

for some $\varepsilon > 0$, $\mathsf{Extract}()$ fails with probability $\mathcal{O}(f_0^{-h+2})$ over all possible choices of the hash functions $H_1, \ldots, H_h$ whenever $f \leq f_0$.

Theorem 2 indicates that the number of cells in the IBF has to be at least χ_h times larger than the number of inserted elements in order to be able to perform full extraction. Fortunately, χ_h is close to 1 for different choices of h. Table 2 gives the values for χ_h as computed in [28].

Table 2. Thresholds for IBF overhead for different number of hash functions from [28].

h	3	4	5	6	7
χ_h	1.222	1.295	1.425	1.570	1.721

Theorem 2 describes the probability of being able to extract all elements from the IBF, but it can not be used to estimate the number of extracted elements when the procedure $\mathsf{Extract}()$ fails, i.e. when $r_{\mathsf{Extract}} < 1$. We look at proof of Theorem 2 to see why it is so.

Proof of Theorem 2. Rephrased from [28]. We first show that an IBF can be described as an (undirected) hypergraph. Let the cells c_i, $i \in [\beta]$, of an IBF correspond to vertices in a hypergraph, and every inserted element x corresponds to a hyperedge

$$e_x = \{\mathsf{c}_i : \forall j \in [h], i = H_j(x)\}$$

between the vertices. An edge e_x corresponds to the element x being inserted into the corresponding cells c_i where $i = H_j(x)$ for all $j \in [h]$. Such a hypergraph where the cardinality of every hyperedge is h, is also called h-uniform hypergraph.

Due to Lemma 1, we can assume that every element is inserted only once into the IBF. Similarly to Algorithm 5, in order to extract an element from the

hypergraph representation, there must exist a vertex with degree 1. If there is such vertex, the hyperedge corresponding to that vertex is also removed. The process continues until there are no more vertices of degree 1 left in the hypergraph.

The extraction succeeds when the degree of all vertices in the end is zero. Otherwise, the degrees of the vertices in the resulting hypergraph must be at least 2. A maximum sub-hypergraph where the minimum degree of every vertex is larger or equal to 2 is called a 2-core of a hypergraph. Thus, the extraction process fails if the 2-core of the hypergraph is non-empty.

In other words, the extraction failure probability corresponds to the probability of finding a 2-core from h-uniform hypergraph. This is well-studied graph-theoretic problem and is considered for example in [16] and [51].

Let $f \leq f_0$ for some f_0. We look at the probability that j hyperedges use only $jh/2$ vertices. There are

$$\binom{f}{j}$$

choices for choosing j hyperedges out of f and

$$\binom{\beta}{jh/2}$$

choices for choosing the corresponding $jh/2$ vertices out of β vertices. For every vertex, the probability for choosing the vertex in a 2-core is $\frac{jh/2}{\beta}$. Thus, we obtain that the probability is at most

$$\binom{f}{j}\binom{\beta}{jh/2}\left(\frac{jh}{2\beta}\right)^{jh}. \tag{2.5}$$

Expression (2.5) can be upper bounded by

$$\left(\frac{f}{e}\right)^{j}\left(\frac{2\beta}{jh}\right)^{jk/2}\left(\frac{jh}{2\beta}\right)^{jh} = \frac{f^{j}}{\beta^{jk/2}}\left(\frac{jhe}{\beta}\right)^{jh/2}\left(\frac{e}{j}\right)^{j}. \tag{2.6}$$

If h is constant and $\beta > (\chi+\varepsilon)f$ then the sum of (2.6) over all $2 \leq j \leq f$ is dominated by $j = 2$. Thus, the extraction failure is dominated by a case of having two hyperedges which share the same vertices. This leads to the failure probability

$$O(f^{-h+2})$$

as given in the theorem statement. □

Summarizing the proof, the method for estimating the failure probability of an IBF is to reduce the probability computation to the probability of finding a non-empty 2-core in a hypergraph. However, this reduction does not provide any information about the corresponding 2-core. For partial extraction, in the hypergraph terms, we need to estimate the probability of having at least one 2-core where the sum of size of the 2-cores is fixed. In our review of the current state of the art, we have not found solution to this problem.

2.7.3. Minimizing IBF overhead

Considering the hypergraph representation of an IBF, there are two main directions to reduce the required overhead for ensuring full extractability with high probability. The first direction is to use non-uniform hypergraphs [44, 58]. In non-uniform hypergraphs, the cardinalities of the hyperedges are not constant and depend on the hyperedge. In the IBF terminology, this means that the number of hash functions depends on the element to be inserted into the IBF. With unbounded hyperedge cardinality, it is possible to achieve IBF overhead arbitrary close to 1 [44] and with bounded hyperedge cardinality the overhead can be reduced to approximately 1.08 [58].

Another direction is to use a family of h-uniform hypergraphs which is given by a probability distribution on the hyperedges. This approach is further studied in [17] and allows to reach IBF overhead close to 1 ($\chi_3 \approx 1.09$, $\chi_4 \approx 1.02$, $\chi_5 \approx 1.01$, $\chi_6 \approx 1.00$, $\chi_7 \approx 1.00$).

2.7.4. Examples of IBF procedures

We present an example for extracting elements from an IBF in Example 6. Additionally, to illustrate the false-positive output of procedure $\mathsf{Test}()$ and failure of procedure $\mathsf{Extract}()$ to extract all elements, we give Examples 7 and 8.

Example 6. Let the field where the elements are sampled from be $\mathbb{F}_{256}$. For convenience, we associate numbers $0,\ldots,255$ with the corresponding field elements in the vector form, where a binary vector is associated with the corresponding binary representation of the number.

Table 3. Hash function values for elements in Example 6

x	$H_1(x)$	$H_2(x)$	$H_3(x)$	$H_4(x)$
13	4	8	15	16
24	1	10	13	17
98	1	6	12	17
124	1	8	13	20
136	2	10	13	17
161	4	8	15	18
166	3	8	11	18
167	1	7	12	20
175	4	10	13	17
198	2	6	14	19
199	5	10	14	19
232	5	6	14	19

Let the elements be

$$\mathcal{S} = \{13, 24, 98, 124, 136, 161, 166, 167, 175, 198, 199, 232\}.$$

We initialize IBF $\mathcal{F}$ with $h = 4$ hash functions and $\beta = 20$ cells. The hash values

for the elements are given as Table 3. The values of the cells of $\mathcal{F}$ after inserting all elements in $\mathcal{S}$ into it are given as Table 4.

Table 4. IBF $\mathcal{F}$ in Example 6

i	$\mathtt{c}_i.\mathtt{count}$	$\mathtt{c}_i.\mathtt{val}$	$\mathtt{c}_i.\mathtt{ch}$
1	4	$24+98+124+167$	$H_{\text{ch}}(24)+H_{\text{ch}}(98)+H_{\text{ch}}(124)+H_{\text{ch}}(167)$
2	2	$136+198$	$H_{\text{ch}}(136)+H_{\text{ch}}(198)$
3	1	166	$H_{\text{ch}}(166)$
4	3	$13+161+175$	$H_{\text{ch}}(13)+H_{\text{ch}}(161)+H_{\text{ch}}(175)$
5	2	$199+232$	$H_{\text{ch}}(199)+H_{\text{ch}}(232)$
6	3	$98+198+232$	$H_{\text{ch}}(98)+H_{\text{ch}}(198)+H_{\text{ch}}(232)$
7	1	167	$H_{\text{ch}}(167)$
8	4	$13+124+161+166$	$H_{\text{ch}}(13)+H_{\text{ch}}(124)+H_{\text{ch}}(161)+H_{\text{ch}}(166)$
9	0		
10	4	$24+136+175+199$	$H_{\text{ch}}(24)+H_{\text{ch}}(136)+H_{\text{ch}}(175)+H_{\text{ch}}(199)$
11	1	166	$H_{\text{ch}}(166)$
12	2	$98+167$	$H_{\text{ch}}(98)+H_{\text{ch}}(167)$
13	4	$24+124+136+175$	$H_{\text{ch}}(24)+H_{\text{ch}}(124)+H_{\text{ch}}(136)+H_{\text{ch}}(175)$
14	3	$198+199+232$	$H_{\text{ch}}(198)+H_{\text{ch}}(199)+H_{\text{ch}}(232)$
15	2	$13+161$	$H_{\text{ch}}(13)+H_{\text{ch}}(161)$
16	1	13	$H_{\text{ch}}(13)$
17	4	$24+98+136+175$	$H_{\text{ch}}(24)+H_{\text{ch}}(98)+H_{\text{ch}}(136)+H_{\text{ch}}(175)$
18	2	$161+166$	$H_{\text{ch}}(161)+H_{\text{ch}}(166)$
19	3	$198+199+232$	$H_{\text{ch}}(198)+H_{\text{ch}}(199)+H_{\text{ch}}(232)$
20	2	$124+167$	$H_{\text{ch}}(124)+H_{\text{ch}}(167)$

In order to extract an element from an IBF cell, only a single element should be contained in the `val` field. We observe in Table 4 that such cells are $\mathtt{c}_3, \mathtt{c}_7, \mathtt{c}_{11}, \mathtt{c}_{16}$ and thus the extractable elements in the first loop are 13, 166 and 167.

The cells of $\mathcal{F}$ after the first extraction round are given as Table 5. As after extracting the elements in the first loop, new cells now contain only a single element. Such cells are $\mathtt{c}_{12}, \mathtt{c}_{15}, \mathtt{c}_{18}, \mathtt{c}_{20}$ and they allow to extract the elements 98, 124 and 161.

By continuing in the same manner, we remove elements 24 and 175 in the third loop, 136 in the fourth loop, 198 and 199 in the fifth loop and 232 in the sixth loop. After the sixth loop no cell in $\mathcal{F}$ contain any element and thus the extraction succeeded. ■

Example 7. We continue with the setting of Example 6. Let there additionally be

Table 5. Non-zero cells of IBF $\mathcal{F}$ in Example 6 after the first extraction loop

i	$\mathtt{c}_i.\mathtt{count}$	$\mathtt{c}_i.\mathtt{val}$	$\mathtt{c}_i.\mathtt{ch}$
1	3	$24+98+124$	$H_{\text{ch}}(24)+H_{\text{ch}}(98)+H_{\text{ch}}(124)$
2	2	$136+198$	$H_{\text{ch}}(136)+H_{\text{ch}}(198)$
4	2	$161+175$	$H_{\text{ch}}(161)+H_{\text{ch}}(175)$
5	2	$199+232$	$H_{\text{ch}}(199)+H_{\text{ch}}(232)$
6	3	$98+198+232$	$H_{\text{ch}}(98)+H_{\text{ch}}(198)+H_{\text{ch}}(232)$
8	2	$124+161$	$H_{\text{ch}}(124)+H_{\text{ch}}(161)$
10	4	$24+136+175+199$	$H_{\text{ch}}(24)+H_{\text{ch}}(136)+H_{\text{ch}}(175)+H_{\text{ch}}(199)$
12	1	98	$H_{\text{ch}}(98)$
13	4	$24+124+136+175$	$H_{\text{ch}}(24)+H_{\text{ch}}(124)+H_{\text{ch}}(136)+H_{\text{ch}}(175)$
14	3	$198+199+232$	$H_{\text{ch}}(198)+H_{\text{ch}}(199)+H_{\text{ch}}(232)$
15	1	161	$H_{\text{ch}}(161)$
17	4	$24+98+136+175$	$H_{\text{ch}}(24)+H_{\text{ch}}(98)+H_{\text{ch}}(136)+H_{\text{ch}}(175)$
18	1	161	$H_{\text{ch}}(161)$
19	3	$198+199+232$	$H_{\text{ch}}(198)+H_{\text{ch}}(199)+H_{\text{ch}}(232)$
20	1	124	$H_{\text{ch}}(124)$

an element 55 with $H_1(55)=4$, $H_2(55)=6$, $H_3(55)=13$ and $H_4(55)=17$. Then for $\mathcal{F}$ from Example 7 the output of $\mathsf{Test}(\mathcal{F},55)$ would be `true` as $\mathtt{c}_4.\mathtt{count}=3$, $\mathtt{c}_6.\mathtt{count}=3$, $\mathtt{c}_{13}.\mathtt{count}=4$ and $\mathtt{c}_{17}.\mathtt{count}=4$ even though the element has not been inserted into $\mathcal{F}$. ■

Example 8. Again, we continue with the setting of Example 6 and with an element 55 from Example 7. If this element is additionally inserted into $\mathcal{F}$, then after third iteration loop there are no elements available to extract as seen in Table 6. The set of extracted elements is $\mathcal{S}_{\mathcal{F}}=\{13,24,98,124,161,166,167\}$ and thus the extraction rate is $r_{\mathsf{Extract}}=7/13$.

Table 6. Non-zero cells of IBF $\mathcal{F}$ in Example 8 after the third extraction round

i	$\mathtt{c}_i.\mathtt{count}$	$\mathtt{c}_i.\mathtt{val}$	$\mathtt{c}_i.\mathtt{ch}$
2	2	$136+198$	$H_{\text{ch}}(136)+H_{\text{ch}}(198)$
4	2	$55+175$	$H_{\text{ch}}(55)+H_{\text{ch}}(175)$
5	2	$199+232$	$H_{\text{ch}}(199)+H_{\text{ch}}(232)$
6	3	$55+198+232$	$H_{\text{ch}}(55)+H_{\text{ch}}(198)+H_{\text{ch}}(232)$
10	3	$136+175+199$	$H_{\text{ch}}(136)+H_{\text{ch}}(175)+H_{\text{ch}}(199)$
13	3	$55+136+175$	$H_{\text{ch}}(55)+H_{\text{ch}}(136)+H_{\text{ch}}(175)$
14	3	$198+199+232$	$H_{\text{ch}}(198)+H_{\text{ch}}(199)+H_{\text{ch}}(232)$
17	3	$55+136+175$	$H_{\text{ch}}(55)+H_{\text{ch}}(136)+H_{\text{ch}}(175)$
19	3	$198+199+232$	$H_{\text{ch}}(198)+H_{\text{ch}}(199)+H_{\text{ch}}(232)$

■

2.8. Data distribution problem in star and complete networks

In this section, we describe the index coding problem which considers data distribution in a network described by a star graph and data exchange problem which considers data distribution in a network described by a complete graph.

We assume that the network is described by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V} = (v_1, \ldots, v_u)$ and some set of edges $\mathcal{E}$ as defined in the subsequent sections. Let $\mathbf{x} = (x_1, \ldots, x_k) \in \mathbb{F}^k$ be an information vector over some field $\mathbb{F}$ of length k. Every node v_i possesses some side information described by a set of indices $\mathcal{Z}_{v_i} \subseteq [k]$, where if the node possesses the element $x_j \in \mathbf{x}$, then $j \in \mathcal{Z}_{v_i}$. We denote the size of $\mathcal{Z}_{v_i}$ as $k_{v_i} \triangleq |\mathcal{Z}_{v_i}|$. Every node is also interested in receiving some elements after the protocol run. The set of requested elements are described by a set $\mathcal{T}_{v_i} \subseteq [k]$, where if node requests x_j, then $j \in \mathcal{T}_{v_i}$.

2.8.1. Index coding problem

Assume that the network consists of a single source node s and τ sink nodes $\mathcal{W}$. Only the source node can transmit to the sink nodes, i.e. the network topology corresponds to a star graph. This means that the set of edges $\mathcal{E}$ is defined as

$$\mathcal{E} \triangleq \{(s,t) : t \in \mathcal{W}\}.$$

We assume that the length of the information vector in this setting is exactly τ and the source node possesses all information, i.e. $\mathcal{Z}_s = [\tau]$.

Every sink node t_i possesses a subset of the information vector, excluding the element $x_i \in \mathbf{x}$ and requests the element $x_i \in \mathbf{x}$. Thus we have $\mathcal{T}_{t_i} = \{i\}$ and $\mathcal{Z}_{t_i} \subsetneq [\tau]$, $i \notin \mathcal{Z}_{t_i}$.

The index coding problem is a problem of finding optimal transmissions from the source such that every sink node can decode their request. As this is a data distribution problem, then we assume that the transmitter knows the corresponding possession indices $\mathcal{Z}_{t_i}$ for every $t_i \in \mathcal{W}$.

In order to describe a protocol for index coding, we firstly assume that the field where the elements are sampled from is binary field $\mathbb{F}_2$ and later generalize the approach to any extension field of $\mathbb{F}_2$.

In order to solve the index coding problem, we use a side information graph as defined in [2].

Definition 1. Given sink nodes $t_i \in \mathcal{W}$, $i \in [\tau]$, and corresponding has-sets described by $\mathcal{Z}_{v_i}$, the side information graph $\mathcal{G}_{\mathtt{SI}} \triangleq (\mathcal{V}_{\mathtt{SI}}, \mathcal{E}_{\mathtt{SI}})$ is a graph consisting of vertices $\mathcal{V}_{\mathtt{SI}} \triangleq [\tau]$ and edges $\mathcal{E}_{\mathtt{SI}} \triangleq \{(i,j) : j \in \mathcal{Z}_{t_i}\}$.

In plain terms, a side information graph has a corresponding vertex for every sink node and has a directed edge from node i to node j if sink node t_i has bit x_j.

In the following definitions, we define a minimum rank of the side information graph.

Definition 2. Let $\mathcal{G}_{\mathtt{SI}} = (\mathcal{V}_{\mathtt{SI}}, \mathcal{E}_{\mathtt{SI}})$ be a graph with a vertex set $\mathcal{V}_{\mathtt{SI}} = [\tau]$. We say that a $\tau \times \tau$-dimensional binary matrix $\mathbf{M}$ fits $\mathcal{G}_{\mathtt{SI}}$ if for all $i \in [\tau]$ we have $(\mathbf{M})_{i,i} = 1$, and for all $i, j \in [\tau]$ we have $(\mathbf{M})_{i,j} = 0$ if there is no edge (i, j) in $\mathcal{E}_{\mathtt{SI}}$.

Definition 3. The minimum rank of graph $\mathcal{G}_{\mathtt{SI}}$ is

$$\text{min-rank}_2(\mathcal{G}_{\mathtt{SI}}) \triangleq \min_{\substack{\mathbf{M} \in \mathbf{S}^{\mathbb{F}}_{\tau \times \tau} \\ \mathbf{M} \text{ fits } \mathcal{G}_{\mathtt{SI}}}} \text{rank}_2(\mathbf{M}).$$

In [2], the authors showed that the optimal number of transmissions of an index coding problem is given by a minimum rank of the corresponding side information graph. The result is referred to in Theorem 3.

Theorem 3 ([2, Theorem 1])**.** *Consider an instance of index coding problem represented by a side information graph $\mathcal{G}_{SI}$. The minimum number of transmissions in any solution for this instance using linear code is given by* $\text{min-rank}_2(\mathcal{G}_{SI})$.

The proof of Theorem 3 is constructive as it also describes the $\text{min-rank}_2(\mathcal{G}_{\mathtt{SI}})$ encoding functions $\mathcal{C}_s$ for the source node and a single decoding function D_{t_i} for every sink node t_i to recover its requested bit x_i. In order to define the encoding and decoding functions, we look at one of the matrices $\mathbf{M}$ which minimizes the rank of the side information graph of the index coding instance, i.e.

$$\text{rank}_2(\mathbf{M}) = \text{min-rank}_2(\mathcal{G}_{\mathtt{SI}}). \tag{2.7}$$

We denote $\pi \triangleq \text{min-rank}_2(\mathcal{G}_{\mathtt{SI}})$ and assume that the first π rows of $\mathbf{M}$ span the whole vector space rowspace($\mathbf{M}$). This means that there exist coefficients $\lambda_{i,j} \in \mathbb{F}_2$, $i \in [\tau]$, $j \in [\pi]$, such that for every $i \in [\tau]$

$$\mathbf{M}^{[i]} = \sum_{j=1}^{\pi} \lambda_{i,j} \mathbf{M}^{[j]}. \tag{2.8}$$

Encoding function: The encoding function E is defined as

$$E(\mathbf{x}) \triangleq (\mathbf{M}^{[i]} \cdot \mathbf{x})_{i \in [\pi]} \tag{2.9}$$

for a matrix $\mathbf{M}$ such that (2.7) holds and where $\mathbf{x} \in \mathbb{F}^{\tau}$ is an information vector. We denote

$$\begin{aligned} \mathbf{b} &\triangleq E(\mathbf{x}) \\ &= (b_1, \ldots, b_\pi) \end{aligned} \tag{2.10}$$

as a vector to be transmitted.

Decoding functions: For a matrix $\mathbf{M}$ which is used to define the encoding function (2.9), let the coefficients $\lambda_{i,j}$ be such that (2.8) holds. Given the received vector $\mathbf{b}$, the decoding function for sink node t_i is defined as

$$D_{t_i}(\mathcal{Z}_{t_i}, \mathbf{b}) \triangleq \sum_{j=1}^{\pi} \lambda_{i,j} b_j - \sum_{j \in \mathcal{Z}_{t_i}} (\mathbf{M})_{i,j} x_j \tag{2.11}$$

We see that the decoding functions are correctly defined in Lemma 4.

Lemma 4. *Given an index coding instance represented by a side information graph $\mathcal{G}_{SI}$ and a corresponding matrix* $\mathbf{M}$ *which fits $\mathcal{G}_{SI}$, the decoding functions D_{t_i} in* (2.11) *outputs x_i if the input vector* $\mathbf{b}$ *to D_{t_i} is obtained as* (2.10) *by encoding* $\mathbf{x}$.

Proof. From (2.9), we have $b_i = \mathbf{M}^{[i]} \cdot \mathbf{x}$, $i \in [\pi]$. Due to the choice of $\mathbf{M}$, we have from (2.8) that $\mathbf{M}^{[i]} = \sum_{j=1}^{\pi} \lambda_{i,j} \mathbf{M}^{[j]}$ and thus

$$\mathbf{M}^{[i]} \cdot \mathbf{x} = \sum_{j=1}^{\pi} \lambda_{i,j} b_j. \tag{2.12}$$

Additionally, we consider a vector

$$c_i \triangleq \mathbf{M}^{[i]} - \mathbf{e}_i. \tag{2.13}$$

As $\mathbf{M}$ fits the side information graph $\mathcal{G}_{\mathtt{SI}}$, then the only non-zero elements in c_i correspond to the bits which sink node t_i possesses. Thus, we have

$$c_i \cdot \mathbf{x} = \sum_{j=1}^{\tau} (\mathbf{M})_{i,j} x_j. \tag{2.14}$$

However, as $(\mathbf{M})_{i,j} = 0$ if $j \notin \mathcal{Z}_{t_i}$, (2.14) can be computed only using elements indexed by $\mathcal{Z}_{t_i}$:

$$c_i \cdot \mathbf{x} = \sum_{j \in \mathcal{Z}_{t_i}} (\mathbf{M})_{i,j} x_j. \tag{2.15}$$

Finally, we can replace (2.12) and (2.15) into right hand side of (2.11) and then use definition (2.13):

$$\begin{aligned} D_{t_i}(\mathcal{Z}_{t_i}, \mathbf{b}) &= (\mathbf{M}^{[i]} \cdot \mathbf{x}) - (c_i \cdot \mathbf{x}) \\ &= (\mathbf{M}^{[i]} - \mathbf{M}^{[i]} + \mathbf{e}_i) \cdot \mathbf{x} = \mathbf{e}_i \cdot \mathbf{x} = x_i \end{aligned}$$

□

So far, we have described the construction in a case where the elements are from $\mathbb{F}_2$, i.e. a single bit. The construction can be generalized to arbitrary n-bit elements by applying the encoding and decoding functions to the corresponding bits of the element. By using this generalization, the source node instead possesses the vector $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_\tau) \in \mathbb{F}_{2^n}^{\tau}$ and obtains the packet $\mathbf{p}$ to be transmitted as

$$E(\mathcal{S}_s) = ((\mathbf{M}^{[j]} \cdot \mathbf{x}_\ell)_{j \in [\pi]})_{\ell \in [n]}. \tag{2.16}$$

The requested element is decoded as

$$D_{t_i}(\mathcal{Z}_{t_i}, \mathbf{p}) = \left(\sum_{j=1}^{\pi} \lambda_{i,j} (\mathbf{p}_\ell)_j - \sum_{j \in \mathcal{Z}_{t_i}} (\mathbf{M})_{i,j} (\mathbf{x}_j)_\ell \right)_{\ell \in [n]}. \tag{2.17}$$

We now have all the tools for describing fully a protocol for data distribution in a network represented by a star graph. The detailed description is given in Protocol 2.

Protocol 2 Data distribution in star graph

Label
Π_2

Network topology
Single source node s. τ sink nodes $\mathcal{W} = \{t_1, \ldots, t_\tau\}$. Edges $\mathcal{E} = \{(s, t_i) : t_i \in \mathcal{W}\}$.

Randomness model
Deterministic

Input
Information vector $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_\tau)$. Every sink node possesses side information described by $\mathcal{Z}_{t_i} \subsetneq [\tau]$, $i \notin \mathcal{Z}_{t_i}$

Goal
Every sink node has $\mathbf{x}_i$.

Offline phase
The oracle uses side information graph $\mathcal{G}_{\mathtt{SI}}$ to define the encoding function E as in (2.16) and decoding functions D_{t_i} as in (2.17).

Online phase

$r = 1$

Computation phase

1. Source computes packet $\mathbf{p} = E(\mathcal{S}_s)$

Transmission phase

2. Source transmits $\mathbf{p}$ to all sink nodes.

Recovery phase

3. Every sink node s_i computes $\mathbf{x}_i = D_{t_i}(\mathcal{Z}_{t_i}, \mathbf{p})$.

Example 9. In Figure 7, we have illustrated an instance of index coding. There is a single transmitter s and sink nodes $\mathcal{W} = \{t_1, t_2, t_3, t_4\}$. The transmitter possesses bits x_1, x_2, x_3 and x_4. Every sink node $t_i \in \mathcal{W}$ has bit $x_{i+1 \pmod 4}$ and requests x_i, i.e. $\mathcal{Z}_{v_i} = \{i+1 \pmod 4\}$ and $\mathcal{T}_{v_i} = \{i\}$.

There are four independent requests, but the source node can satisfy all requests by transmitting only three field elements in a packet $\mathbf{p} = (x_1 + x_2, x_2 + x_3, x_3 + x_4)$. Nodes t_1, t_2, and t_3 can recover x_i by subtracting x_{i+1} from received $x_i + x_{i+1}$. Node t_4 can recover x_4 by adding received packets and subtracting x_1, i.e.

$$(x_1 + x_2) + (x_2 + x_3) + (x_3 + x_4) - x_1,$$

considering that operations are performed over $\mathbb{F}_2$.

We show that this transmission schedule is indeed optimal. For that, we first illustrate the corresponding side information graph $\mathcal{G}_{\mathtt{SI}}$ in Figure 8. The graph $\mathcal{G}_{\mathtt{SI}}$ consists of nodes $\mathcal{V}_{\mathtt{SI}} = \{1, 2, 3, 4\}$ and edges $\mathcal{E}_{\mathtt{SI}} = \{(1,2), (2,3), (3,4), (4,1)\}$.

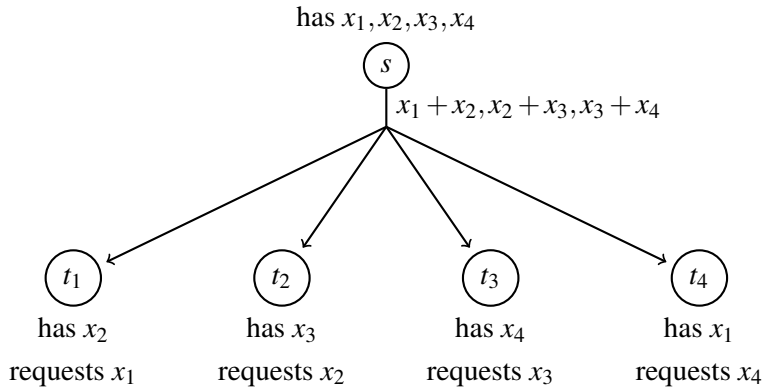

Figure 7. Example of index coding

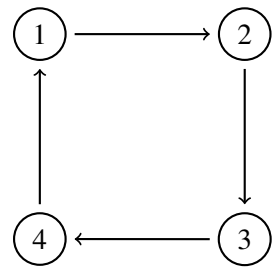

Figure 8. Side information graph corresponding to Example 9

An example matrix which fits the side information graph is

$$\mathbf{M} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}. \tag{2.18}$$

We observe that $\text{rank}_2(\mathbf{M}) = 3$ and by enumerating over all matrices which fit $\mathcal{G}_{\mathtt{SI}}$ that $\mathbf{M}$ indeed minimizes $\text{min-rank}_2(\mathcal{G}_{\mathtt{SI}})$. Thus, the matrix $\mathbf{M}$ also defines optimal transmission schedule.

For matrix $\mathbf{M}$ as given in (2.18), first three rows are linearly independent and can be used to define the encoding function. This means that the coefficients $\lambda_{i,j}$ from (2.8) are

$$\begin{array}{lll} \lambda_{1,1} = 1, & \lambda_{1,2} = 0, & \lambda_{1,3} = 0, \\ \lambda_{2,1} = 0, & \lambda_{2,2} = 1, & \lambda_{2,3} = 0, \\ \lambda_{3,1} = 0, & \lambda_{3,2} = 0, & \lambda_{3,3} = 1, \\ \lambda_{4,1} = 1, & \lambda_{4,2} = 1, & \lambda_{4,3} = 1. \end{array}$$

By using the encoding function definition (2.9), we get

$$E(\mathbf{x}) = ((1,1,0,0)\cdot\mathbf{x}, (0,1,1,0)\cdot\mathbf{x}, (0,0,1,1)\cdot\mathbf{x}).$$

From (2.11) we get the decoding functions

$$\begin{aligned}
D_{v_1}(\mathcal{Z}_{v_1}, \mathbf{b}) &= b_1 - x_2, \\
D_{v_2}(\mathcal{Z}_{v_2}, \mathbf{b}) &= b_2 - x_3, \\
D_{v_3}(\mathcal{Z}_{v_3}, \mathbf{b}) &= b_3 - x_4, \\
D_{v_4}(\mathcal{Z}_{v_4}, \mathbf{b}) &= b_1 + b_2 + b_3 - x_1.
\end{aligned}$$

■

2.8.2. Data exchange problem

Let there be u nodes $\mathcal{V} = \{v_1, \ldots, v_u\}$ where every node v_i possesses arbitrary side information information described by indices $\mathcal{Z}_{v_i} \subseteq [k]$, and requests the elements missing in its side information, i.e. $\mathcal{T}_{v_i} = [k] \setminus \mathcal{Z}_{v_i}$.

The underlying graph of the network is a complete graph, i.e. for every node v_i there are edges (v_i, v_j) for every $v_j \in \mathcal{V} \setminus \{v_i\}$. We consider the data exchange problem as a case of the data distribution problem. Thus, we assume that there exists an oracle which knows the indices of the elements of current has-sets of every node.

In [21], a solution to the data exchange problem was given using a matrix rank minimization approach. In order to describe the protocol, we introduce an additional representation for the has-sets of the nodes. We associate with every node $v_i \in \mathcal{V}$ a set of $k_{v_i} \times k$-dimensional matrices $\mathbf{P}_{v_i} \subseteq \mathbf{S}^{\mathbb{F}}_{k_{v_i} \times k}$:

$$\mathbf{P}_{v_i} \triangleq \{\mathbf{M}_{v_i} \in \mathbf{S}^{\mathbb{F}}_{k_{v_i} \times k} : \forall \ell \in [k_{v_i}], (\mathbf{M}_{v_i})_{\ell, j} = 0 \text{ if } j \notin \mathcal{Z}_{v_i}\} \tag{2.19}$$

Otherwise, if $j \in \mathcal{Z}_{v_i}$, then $(\mathbf{M}_{v_i})_{\ell, j}$ can take any value in the underlying field $\mathbb{F}$.

We denote the set of $\left(\sum_{v_i \in \mathcal{V}} k_{v_i}\right) \times k$-dimensional matrices $\mathbf{P} \subseteq \mathbf{S}^{\mathbb{F}}_{k_{v_i} \times k}$ which is concatenation of matrices in matrix sets $\mathbf{P}_{v_i}$:

$$\mathbf{P} \triangleq \left\{ \begin{bmatrix} \mathbf{M}_{v_1} \\ \vdots \\ \mathbf{M}_{v_u} \end{bmatrix} : \mathbf{M}_{v_i} \in \mathbf{P}_{v_i} \right\}. \tag{2.20}$$

For every node v_i, we fix a matrix $\mathbf{N}_{v_i}$ from the set of matrices $\mathbf{P}_{v_i}$ where in the j-th row the vector induced by columns $\mathcal{Z}_{v_i}$ form j-th unit vector.

In [21], the following Theorem 5 establishes the optimal number of packets to be exchanged such that all the nodes can recover the elements in $\mathcal{S}$.

Theorem 5 ([21, Theorem 5])**.** *The minimum number of transmissions π achieved by linear codes is given by*

$$\pi = \min_{\mathbf{M} \in \mathbf{P}} \operatorname{rank}(\mathbf{M}) \tag{2.21}$$

subject to constraints

$$\operatorname{rank}\left(\begin{bmatrix}\mathbf{M}\\ \mathbf{N}_{v_i}\end{bmatrix}\right) = k, \quad \forall v_i \in \mathcal{V}. \tag{2.22}$$

Even though Theorem 5 was given without a proof, it can be shown that the proof is constructive, giving specific encoding and decoding functions for the data exchange problem. Let $\mathbf{M}$ be the matrix which satisfies the optimization problem in (2.21) such that there are exactly π non-zero rows. Let $\mathbf{M}_{v_i}$, $v_i \in \mathcal{V}$, be the corresponding partition of $\mathbf{M}$ as given in (2.20).

Encoding functions: Every node v_i encodes the packets $\mathbf{p}_{v_i}$ to be transmitted as

$$E_{v_i}(\mathcal{Z}_{v_i}) \triangleq \mathbf{M}_{v_i}\mathbf{x}^\top. \tag{2.23}$$

Additionally, when the j-th row $(\mathbf{M}_{v_i})^{[j]}$ is $\mathbf{0}$, then transmission of the packet is omitted.

Decoding functions: From the received packets $\mathbf{p}_{v_i}$, $v_i \in \mathcal{V}$, we construct a single packet by considering packets $\mathbf{p}_{v_i}$ as a single-column matrices:

$$\mathbf{p}' \triangleq \begin{bmatrix}(\mathbf{p}_{v_1})^\top\\ \vdots \\ (\mathbf{p}_{v_u})^\top\end{bmatrix}. \tag{2.24}$$

We additionally consider a vector $\mathbf{x}_{v_i}$ which is induced from $\mathbf{x}$ by the columns $\mathcal{Z}_{v_i}$.

Node v_i can recover elements in the request set by solving the system of linear equations

$$\begin{bmatrix}\mathbf{M}\\ \mathbf{N}_{v_i}\end{bmatrix}\mathbf{X} = \begin{bmatrix}\mathbf{p}'\\ (\mathbf{x}_{v_i})^\top\end{bmatrix}$$

for an unknown variable $\mathbf{X}$.

By denoting the function for solving a system of linear equations as $\mathsf{LinSolve}()$, we get the decoding function:

$$D_{v_i}(\mathcal{Z}_{v_i}, \mathbf{p}') \triangleq \mathsf{LinSolve}\left(\begin{bmatrix}\mathbf{M}\\ \mathbf{N}_{v_i}\end{bmatrix}, \begin{bmatrix}\mathbf{p}'\\ (\mathbf{x}_{v_i})^\top\end{bmatrix}\right). \tag{2.25}$$

Lemma 6. *Given a data exchange problem instance described by nodes $\mathcal{V}$ and corresponding has-sets described by the indices $\mathcal{Z}_{v_i}$ for every node $v_i \in \mathcal{V}$, a matrix $\mathbf{M}$ which is a solution to an optimization problem in* (2.21) *subject to constraints* (2.22)*, by encoding the packets using encoding functions E_{v_i} as defined in* (2.23) *and decoding the packets using functions D_{v_i} as defined in* (2.25)*, node v_i recovers* $\mathbf{x}$.

We omit the proof of Lemma 6 for now as it follows directly from Theorem 18. We give the description of protocol for the data exchange problem in Protocol 3.

Protocol 3 Data distribution in a complete graph

Label

Π_3

Network topology

u nodes $\mathcal{V}$. Edges $\mathcal{E} = \{(v_i, v_j) : v_i, v_j \in \mathcal{V}, v_i \neq v_j\}$.

Randomness model

Deterministic

Input

Information vector $\mathbf{x} = (x_1, \ldots, x_k)$. Every node $v_i \in \mathcal{V}$ possesses a subset of the information vector described by the indices $\mathcal{Z}_{v_i} \subseteq [k]$.

Goal

Ever node $v_i \in \mathcal{V}$ requests all missing elements $\mathcal{T}_{v_i} = [k] \setminus \mathcal{Z}_{v_i}$.

Offline phase

The oracle finds $\mathbf{M} \in \mathbf{P}$ which minimizes (2.21) such that (2.22) holds for every $v_i \in \mathcal{V}$. For every node $v_i \in \mathcal{V}$, the oracle defines encoding functions E_{v_i} as in (2.23) and decoding function D_{v_i} as in (2.25).

Online phase

$r = 1$

Computation phase

1. Every node v_i computes packet $\mathbf{p}_{v_i} = E_{v_i}(\mathcal{S}_{v_i})$.

Transmission phase

2. Every node v_i transmits $\mathbf{p}_{v_i}$ to all other nodes.

Recovery phase

3. Every node v_i uses received packets $\mathbf{p}_{v_\ell}$, $v_\ell \in \mathcal{V}$ to construct $\mathbf{p}'_{v_i}$ as in (2.24) and obtains missing elements from $D_{v_i}(\mathcal{S}_{v_i}, \mathbf{p}'_{v_i})$.

Example 10. An instance of data exchange problem is illustrated in Figure 9. There are four nodes $\mathcal{V} = \{v_1, v_2, v_3, v_4\}$ in a complete graph. The possessed bits are given by $\mathcal{Z}_{v_1} = \{1\}$, $\mathcal{Z}_{v_2} = \{2,4\}$, $\mathcal{Z}_{v_3} = \{2,3\}$ and $\mathcal{Z}_{v_4} = \{1,2\}$. Every node requests all missing elements.

If node v_2 transmits $x_2 + x_4$, v_3 transmits $x_2 + x_3$ and v_4 transmits $x_1 + x_3$, then we observe that all nodes can recover all elements.

In this complete example, we look at how the solution is obtained and that it is optimal. For this, we first choose matrices $\mathbf{M}_{v_i}$ from $\mathbf{P}_{v_i}$ as defined in (2.19):

$$\mathbf{M}_{v_1} = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}, \qquad \mathbf{M}_{v_2} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$\mathbf{M}_{v_3} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \qquad \mathbf{M}_{v_4} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

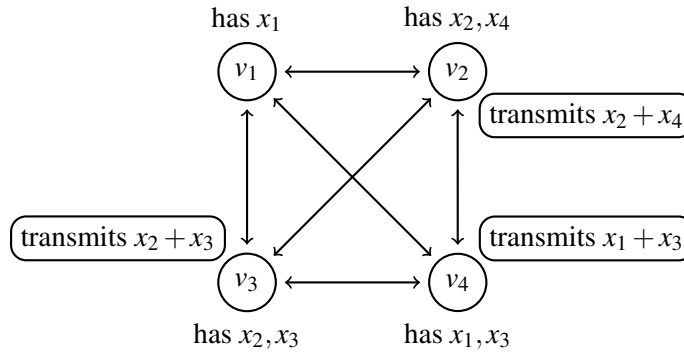

Figure 9. An example of data exchange instance

The corresponding matrix $\mathbf{M}$ is

$$\mathbf{M} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

The following matrices are fixed:

$$\mathbf{N}_{v_1} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}, \qquad \mathbf{N}_{v_2} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$
$$\mathbf{N}_{v_3} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \qquad \mathbf{N}_{v_4} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

First, we see that for every $v_i \in \mathcal{V}$, we have

$$\operatorname{rank}\left(\begin{bmatrix} \mathbf{M} \\ \mathbf{N}_{v_i} \end{bmatrix}\right) = 4.$$

By exhaustive search, we can also verify that such $\mathbf{M}$ has minimum rank 3 over all matrices in $\mathbf{P}$ and thus any protocol transmits at least three elements.

From (2.23), the packets to be transmitted are

$$\begin{aligned} \mathbf{p}_{v_1} &= (0), \\ \mathbf{p}_{v_2} &= (x_2 + x_4, 0), \\ \mathbf{p}_{v_3} &= (x_3 + x_4, 0), \\ \mathbf{p}_{v_4} &= (x_1 + x_3, 0). \end{aligned}$$

Due to the remark, the zeros are omitted from transmissions and the nodes transmit three elements in total.

The concatenated received packet is

$$\mathbf{p}' = (0, x_2 + x_4, 0, x_3 + x_4, 0, x_1 + x_3, 0)^\top$$

and the vectors $\mathbf{x}_{v_i}$ are

$$\begin{aligned} \mathbf{x}_{v_1} &= (x_1), & \mathbf{x}_{v_2} &= (x_2, x_4), \\ \mathbf{x}_{v_3} &= (x_2, x_3), & \mathbf{x}_{v_4} &= (x_1, x_3). \end{aligned}$$

For now, we only look the system of linear equations node v_1 has to solve as it follows analogously for other nodes. Node v_1 solves the system of linear equations for $\mathbf{X}$:

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \mathbf{X} = \begin{bmatrix} 0 \\ x_2 + x_4 \\ 0 \\ x_3 + x_4 \\ 0 \\ x_1 + x_3 \\ 0 \\ x_1 \end{bmatrix}. \tag{2.26}$$

As the matrix on the left-hand side of (2.26) is full rank, then there is a solution. This solution gives the complete information vector $\mathbf{x}$. ■

3. DATA SYNCHRONIZATION USING PARTIALLY EXTRACTABLE INVERTIBLE BLOOM FILTERS

In this chapter, we use an alternative matrix-based representation for an IBF. In Section 3.1.1 we show an alternative condition for extraction failure from an IBF. We use this condition in Section 3.1.2 to present a novel method for analyzing IBF efficiency for all possible parameters even in case of partial extraction. The result is given as Theorem 11. We supplement the theoretical results with experimental simulations in Section 3.1.3.

Then, in Section 3.2.1 we describe a data synchronization protocol using IBFs. We identify a hard requirement on knowing the size of the symmetric difference between the sets to be synchronized. In Section 3.2.2, we present a novel protocol for data synchronization in case the estimate on the size of symmetric difference is unknown or inexact. The result is based on our new method in Section 3.1.2 and is given as Lemma 15.

Finally, in Section 3.2.3, we present theoretical estimates on the number of rounds for the new data synchronization protocol and compare it to numerical simulations.

3.1. Partially extractable invertible Bloom filters

In Section 2.7, invertible Bloom filters were described as an efficient data structure for storing data. However, it is currently unknown how IBFs behave when we aim at extracting only a subset of the inserted elements.

The extraction rate r_{Extract} was defined in (2.4) as the ratio between the number of extracted elements to the total number of inserted elements. In this section, we estimate the probability of having extraction rate

$$r_{\mathsf{Extract}} < 1.$$

3.1.1. IBF state matrix representation

Extracting elements from an IBF is similar to iterative decoding of *low-density parity-check* (LDPC) codes. Gallager introduced LDPC codes and iterative decoding algorithm in [24]. LDPC codes can be used to reconstruct codewords sent over an erasure channel. By associating the elements which are inserted into an IBF with variable nodes of the Tanner graph of the parity-check matrix, and by associating the IBF cells with the check nodes, the $\mathsf{Extract}()$ procedure corresponds to codeword decoding procedure of an LDPC code.

Richardson and Urbanke defined a *stopping set* as a subset of check nodes such that there exists no variable node with exactly one edge to the stopping set [57]. If there is a stopping set in a parity-check matrix of an LDPC code, then it is not possible to recover the erasures of the variable nodes corresponding to the stopping set. As in the case of IBFs, the goal is to extract all inserted elements and as such, the existence of any stopping set leads to a failure of the procedure.

Yugawa and Wadayama gave more detailed description of an IBF as a parity check matrix, which they called *state matrix* of an IBF in [64]. We give their description as Definition 4.

Definition 4. The state matrix $\mathbf{F}$ of an IBF $\mathcal{F}$ which contains f elements $x_1, \ldots, x_f$ is a $\beta \times f$ binary matrix which contains $f \cdot h$ non-zero entries, where the value in the cell $(\mathbf{F})_{i,j}$ is one if there exists $\ell \in [h]$, such that $H_\ell(x_j) = i$ and zero otherwise.

We emphasize that as the elements $x_1, \ldots, x_f$ can be arbitrary, then this representation is not sufficient for determining the elements which have been inserted into the IBF. As such, the state matrix representation allows us to determine which cells can be used to extract elements from an IBF. After determining these cells, we have to use the initial IBF where the values of the elements to be extracted are contained in the `val` field of these cells.

Due to (2.3), the outputs of the hash functions are distinct for every $x \in \mathbb{F}$ and thus the Hamming weight of every column is exactly h. Without loss of generality, we take

$$\beta_H \triangleq \frac{\beta}{h}$$

and assume that the output of i-th hash function is in the range

$$[(i-1)\beta_H + 1, i\beta_H].$$

This allows to partition the IBF $\mathcal{F}$ as sub-IBFs $\mathcal{F}_i$, $i \in [h]$, where every element is included in only a single cell. Correspondingly, the matrix $\mathbf{F}$ can be partitioned into $\beta_H \times f$ submatrices $\mathbf{F}_i$ for $i \in [h]$:

$$\mathbf{F} \triangleq \begin{bmatrix} \mathbf{F}_1 \\ \mathbf{F}_2 \\ \vdots \\ \mathbf{F}_h \end{bmatrix}. \tag{3.1}$$

We now consider the set of these $\beta_H \times f$ submatrices, i.e. matrices of column weight one. We denote the set as

$$\mathbf{S}_{\beta_H, f} \triangleq \{\mathbf{M} : \mathbf{M} \text{ is } \beta_H \times f \text{ binary matrix with every column of weight } 1\}.$$

As the number of weight one vectors is β_H and there are f columns, the number of such matrices is

$$(\beta_H)^f.$$

We denote the set of matrices where h state matrices are concatenated as in (3.1) by $\left(\mathbf{S}_{\beta_H, f}\right)^h$. The number of such matrices is

$$(\beta_H)^{hf}. \tag{3.2}$$

We can see that if the i-th row of $\mathbf{F}$ has Hamming weight one, then it corresponds to a `count` field value 1 in the corresponding i-th cell $\mathcal{F}[i]$. Thus, it is possible to extract element x_j from $\mathcal{F}$ corresponding to the j-th column in $\mathbf{F}$ where one appeared. Such a cell $(\mathbf{F})_{i,j}$ is called a pivot.

If the element x_j is removed from $\mathcal{F}$, then in the corresponding state matrix the j-th column is removed. By removing the j-th column, we can now look for another row of weight one in the remaining matrix to extract another value. This process corresponds to the peeling nature of the $\mathsf{Extract}()$ procedure and stops until there are no rows of weight one. As we partitioned the matrix $\mathbf{F}$ into submatrices, then this also means that in every submatrix $\mathbf{F}_i$, $i \in [h]$, we have no row of weight one. The matrix which has no rows of weight one is also called a stopping matrix. The count of stopping matrices in $\mathbf{S}_{\beta_H, f}$ is given by

$$\zeta(\beta_H, f) \triangleq \left|\{\mathbf{M} \in \mathbf{S}_{\beta_H, f} : \mathbf{M} \text{ is a stopping matrix}\}\right|.$$

Example 11. We continue with the setting of Example 6. The corresponding state matrix representation $\mathbf{F}$ of the IBF $\mathcal{F}$ in Example 6 is given as (3.3).

$$\mathbf{F} = \begin{array}{c} \left[\begin{array}{cccccccccccc} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \boxed{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{array}\right] \\ \begin{array}{cccccccccccc} \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow \\ 13 & 24 & 98 & 124 & 136 & 161 & 166 & 167 & 175 & 198 & 199 & 232 \end{array} \end{array} \tag{3.3}$$

We have highlighted the pivots in $\mathbf{F}$ to indicate which cells allow extracting the elements. By considering the columns of the pivots, we see that the elements which can be extracted are 13, 166 and 167. The columns corresponding to the extracted elements can be removed, and therefore in the next iteration new pivot elements could appear. ■

In [64], the authors proposed a recursive equation for counting the number of stopping matrices in $\mathbf{S}_{\beta_H,f}$. Their result is given as Theorem 7.

Theorem 7 ([64, Theorem 1])**.** *The number of stopping matrices in $\mathbf{S}_{\beta_H,f}$ can be expressed as*

$$\zeta(\beta_H, f) = (\beta_H)^f - \sum_{i=1}^{\min(\beta_H,f)} i! \binom{\beta_H}{i} \binom{f}{i} \zeta(\beta_H - i, f - i),$$

for $\beta_H \geq 1$ and $f \geq 1$.

3.1.2. Counting argument for estimating success probability of partial IBF extraction

We can now state the first result which gives us the number of such state matrices from $(\mathbf{S}_{\beta_H,f})^h$ which allow for extraction of at least g elements.

Lemma 8. *For IBFs with h hash functions, β cells and f inserted elements, the number of state matrices allowing to extract at least g elements is*

$$\nu(\beta_H,f,h,g)=\binom{f}{g}\sum_{\substack{\mathbf{b}\in[0,g]^h\\ \sum\mathbf{b}\geq g}}\left(\Psi(g,\mathbf{b})\prod_{b\in\mathbf{b}}\left[\binom{\beta_H}{b}b!\zeta(\beta_H-b,f-b)\right]\right),$$

where $\beta_H=\dfrac{\beta}{h}$ and the function Ψ is defined recursively as

$$\Psi(g,\mathbf{b})=\begin{cases}1 & \text{if } g=0 \text{ and } \mathbf{b}=\mathbf{0}\\ 0 & \text{if } g=0 \text{ and } \mathbf{b}\neq\mathbf{0}\\ \prod_{b\in\mathbf{b}}\binom{g}{b}-\sum_{i=1}^{g-1}\left(\binom{g}{i}\Psi(i,\mathbf{b})\right) & \text{otherwise}\end{cases}. \tag{3.4}$$

Proof. We first consider the case where $h=1$ and then extend to a general case $h\geq 1$.

$h=1$: in order to be able to extract exactly g elements from $\mathbf{F}=\mathbf{F}_1$, there has to exist exactly g rows of weight one. We denote the indices of these rows as $\mathcal{T}$ and the indices of columns where these rows have ones as $\mathcal{U}$. We consider the following non-overlapping submatrices of $\mathbf{F}$:

1. induced by rows-columns pair $(\mathcal{T},\mathcal{U})$ – as every row and column has only a single one, then this is a permutation of the identity matrix $\mathbf{I}$;
2. induced by rows-columns pair $(\mathcal{T},[f]\setminus\mathcal{U})$ – as rows $\mathcal{T}$ have only a single one in every row, then this submatrix must be a zero matrix;
3. induced by rows-columns pair $([\beta_H]\setminus\mathcal{T},\mathcal{U})$ – as every column has only a single one in every row and they were in $\mathcal{T}$, then must be a zero matrix;
4. induced by rows-columns pair $([\beta_H]\setminus\mathcal{T},[f]\setminus\mathcal{U})$ – as it is not possible to extract any further element, then it must be a stopping matrix.

There are

$$\binom{f}{g} \tag{3.5}$$

ways to choose the rows $\mathcal{T}$ and

$$\binom{\beta_H}{g} \tag{3.6}$$

ways to choose columns $\mathcal{U}$. Additionally, in case 1 there are

$$g! \tag{3.7}$$

permutations of $\mathbf{I}$ and in case 4

$$\zeta(\beta_H - g, f - g) \tag{3.8}$$

ways to choose the stopping matrix. For both cases 2 and 3 there is only a single zero matrix. Thus, to obtain the total number of matrices which allow extracting g elements, we have to combine different choices, i.e. (3.5), (3.6), (3.7) and (3.8):

$$\nu_1(\beta_H, f, g) \triangleq \binom{f}{g}\binom{\beta_H}{g} g! z(\beta_H - g, f - g). \tag{3.9}$$

Any h: we consider the partition of $\mathbf{F}$ into submatrices $\mathbf{F}_i$, $i \in [h]$ as given in (3.1).

In order to extract at least g elements, we first can choose the columns where g elements are extracted from. There are

$$\binom{f}{g} \tag{3.10}$$

ways to choose such columns. We denote the indices of these columns as $\mathcal{U}$.

Now, we consider the individual matrices $\mathbf{F}_i$, $i \in [h]$. We analyze how we can extract elements from all such matrices. It is not necessary that all g elements are available for extraction from every matrix $\mathbf{F}_i$, $i \in [h]$. It is sufficient that for every element there exists at least a single matrix where it can be extracted from. By denoting the set of column indices where elements are extracted from matrix $\mathbf{F}_i$ as $\mathcal{U}_i$, then this means that we require

$$\bigcup_{i \in [h]} \mathcal{U}_i = \mathcal{U}.$$

Alternatively, by denoting $b_i = |\mathcal{U}_i|$, $\mathbf{b} = (b_1, \ldots, b_h)$, and the number of ways to choose the columns $\mathcal{U}_i$, $i \in [h]$, as

$$\Psi(g, \mathbf{b}), \tag{3.11}$$

we obtain that the number of state matrices in $\mathbf{S}_{\beta_H, f}$ which allow for extracting b_i elements from $\mathbf{F}_i$ is

$$\nu(\beta_H, f, h, g) = \binom{f}{g} \Psi(g, \mathbf{b}) \prod_{b \in \mathbf{b}} \left[\binom{\beta_H}{b} b! \zeta(\beta_H - b, f - b) \right]$$

by multiplying (3.9) for every $b \in \mathbf{b}$ with (3.10) and (3.11).

To iterate over all such $\mathcal{U}_i$, $i \in [h]$, which allow to extract at least g elements, we denote

$$\mathbf{b} \in [0, g]^h \tag{3.12}$$

such that

$$\sum \mathbf{b} \geq g. \tag{3.13}$$

By summing over all vectors $\mathbf{b}$ where (3.12) and condition (3.13) holds, we have

$$\nu(\beta_H, f, h, g) \triangleq \binom{f}{g} \sum_{\substack{\mathbf{b}\in[0,g]^h \\ \sum \mathbf{b} \geq g}} \left(\Psi(g, \mathbf{b}) \prod_{b\in\mathbf{b}} \left[\binom{\beta_H}{b} b! \zeta(\beta_H - b, f - b) \right] \right).$$

This completes the first part of the lemma statement.

To prove the second part of the statement, we need to show that the function $\Psi(g, \mathbf{b})$ counts the number of choices for columns $\mathcal{U}_i$, $i \in [h]$, such that

$$\bigcup_{i\in[h]} \mathcal{U}_i = \mathcal{U}$$

where $b_i = |\mathcal{U}_i|$ and $\mathbf{b} = (b_1, \ldots, b_h)$.

First, we define $\Psi(g, \mathbf{b}) = 0$ for the case $g = 1$ and $\mathbf{b} = \mathbf{0}$, and $\Psi(g, \mathbf{b}) = 0$ for the case $g = 1$ and $\mathbf{b} \neq \mathbf{0}$. For $g > 0$, we first compute all possible ways to choose every $\mathcal{U}_i \subseteq \mathcal{U}$ where $|\mathcal{U}_i| = b_i$ for $i \in [h]$, which is

$$\prod_{i\in[h]} \binom{g}{b_i}. \tag{3.14}$$

However, these assignments also contain the cases where

$$\left| \cup_{i\in[h]} \mathcal{U}_i \right| = j < g. \tag{3.15}$$

For every $0 \leq j < g$, the number of such assignments is

$$\Psi(j, \mathbf{b})$$

and there are

$$\binom{g}{j}$$

ways to choose $\cup_{i\in[h]} \mathcal{U}_i$ such that (3.15) holds. By summing over all $0 \leq j < g$, we have that the number of unsuitable assignments is

$$\sum_{j=0}^{g-1} \left(\binom{g}{j} \Psi(j, \mathbf{b}) \right). \tag{3.16}$$

By subtracting (3.16) from (3.14), the number of ways to choose the columns $(\mathcal{U}_i)_{i\in[h]}$ becomes

$$\Psi(g, \mathbf{b}) \triangleq \prod_{i\in[h]} \binom{g}{b_i} - \sum_{j=0}^{g-1} \left(\binom{g}{j} \Psi(j, \mathbf{b}) \right).$$

By rewriting the variables, we obtain the recursive function in the lemma statement. □

In order to speed up the computations, we also present a non-recursive form of (3.4) in Corollary 9.

Corollary 9. *Recursive equation* $\Psi(g,\mathbf{b})$ *in Theorem 8 can be written as*

$$\Psi(g,\mathbf{b}) \triangleq \sum_{i=0}^{g}(-1)^{g-i}\binom{g}{i}\prod_{b\in\mathbf{b}}\binom{i}{b} \tag{3.17}$$

for any $g \geq 0$.

Proof. To prove the claim, we use induction. The claim trivially holds for $g = 1$.

Let us assume that the claim holds for all $1 \leq j \leq g$, i.e.

$$\Psi(j,\mathbf{b}) = \sum_{i=0}^{j}(-1)^{j-i}\binom{j}{i}\prod_{b\in\mathbf{b}}\binom{i}{b}. \tag{3.18}$$

We show that the claim holds also for $g+1$:

$$\Psi(g+1,\mathbf{b}) = \prod_{b\in\mathbf{b}}\binom{g+1}{b} - \sum_{j=0}^{g}\binom{g+1}{j}\Psi(j,\mathbf{b}) \tag{3.19}$$

$$= \prod_{b\in\mathbf{b}}\binom{g+1}{b} - \sum_{j=0}^{g}\binom{g+1}{j}\left[\sum_{i=0}^{j}(-1)^{j-i}\binom{j}{i}\prod_{b\in\mathbf{b}}\binom{i}{b}\right] \tag{3.20}$$

$$= \prod_{b\in\mathbf{b}}\binom{g+1}{b} - \sum_{j=0}^{g}\sum_{i=0}^{j}(-1)^{j-i}\binom{g+1}{j}\binom{j}{i}\prod_{b\in\mathbf{b}}\binom{i}{b} \tag{3.21}$$

$$= \prod_{b\in\mathbf{b}}\binom{g+1}{b} - \sum_{i=0}^{g}\sum_{j=0}^{g}(-1)^{j-i}\binom{g+1}{j}\binom{j}{i}\prod_{b\in\mathbf{b}}\binom{i}{b} \tag{3.22}$$

$$= \prod_{b\in\mathbf{b}}\binom{g+1}{b} - \sum_{i=0}^{g}\prod_{b\in\mathbf{b}}\binom{i}{b}\sum_{j=0}^{g}(-1)^{j-i}\binom{g+1}{j}\binom{j}{i} \tag{3.23}$$

$$= \prod_{b\in\mathbf{b}}\binom{g+1}{b} - \sum_{i=0}^{g}\prod_{b\in\mathbf{b}}\binom{i}{b}\sum_{j=0}^{g}(-1)^{j-i}\binom{g+1}{i}\binom{g+1-i}{j-i} \tag{3.24}$$

$$= \prod_{b\in\mathbf{b}}\binom{g+1}{b} - \sum_{i=0}^{g}\binom{g+1}{i}\prod_{b\in\mathbf{b}}\binom{i}{b}\sum_{j=0}^{g}(-1)^{j-i}\binom{g+1-i}{j-i} \tag{3.25}$$

$$= \prod_{b\in\mathbf{b}}\binom{g+1}{b} - \sum_{i=0}^{g}\binom{g+1}{i}\prod_{b\in\mathbf{b}}\binom{i}{b}(-1)^{g-i} \tag{3.26}$$

$$= \prod_{b\in\mathbf{b}}\binom{g+1}{b} + \sum_{i=0}^{g}\binom{g+1}{i}\prod_{b\in\mathbf{b}}\binom{i}{b}(-1)^{g+1-i} \tag{3.27}$$

$$= (-1)^{g+1-(g+1)}\binom{g+1}{g+1}\prod_{b\in\mathbf{b}}\binom{g+1}{b} + \sum_{i=0}^{g}\binom{g+1}{i}\prod_{b\in\mathbf{b}}\binom{i}{b}(-1)^{g+1-i} \tag{3.28}$$

$$= \sum_{i=0}^{g+1} \binom{g+1}{i} \prod_{b \in \mathbf{b}} \binom{i}{b} (-1)^{g+1-i}. \tag{3.29}$$

We get (3.19) from definition (3.4). (3.20) is obtained by applying induction assumption (3.18). (3.21) is obtained by expanding by $\binom{g+1}{j}$. (3.22) is obtained by reordering the indexing variables from $0 \leq j \leq g$ and $0 \leq i \leq j$ to $0 \leq i \leq g$ and $0 \leq j \leq g$. (3.23) is obtained by grouping by $\prod_{b \in \mathbf{b}} \binom{i}{b}$. (3.24) is obtained by applying binomial identity $\binom{n}{m}\binom{m}{k} = \binom{n}{k}\binom{n-k}{m-k}$. (3.25) is obtained by grouping by $\binom{g+1}{i}$. (3.26) is obtained by applying binomial identity $\sum_{k=0}^{n} (-1)^k \binom{n}{k} = 0$. (3.27) is obtained by grouping by -1. (3.28) is obtained by multiplying by $(-1)^{g+1-(g+1)} = 1$ and $\binom{g+1}{g+1} = 1$. (3.29) is obtained by collecting individual addend in the sum variable.

We see that the induction step holds and thus (3.17) holds for any $g \geq 1$. We also see that (3.17) holds for $g = 0$ as $\binom{0}{b} = 0$ for any $b \in \mathbb{N}$. □

The result in Lemma 10 below gives a lower bound for the probability of being able to extract a given ratio of elements from an IBF.

Lemma 10. *Let the IBF be initialized with β cells and h hash functions. After choosing uniformly f elements and inserting these elements to an IBF, we denote the number of elements the procedure* Extract() *returns as a random variable $Y_{\beta_H,h,f}$.*

Then, the probability that the number of extracted elements is at least y is lower bounded by

$$\Pr(Y_{\beta_H,h,f} \geq y) \geq \sum_{g=y}^{f} \frac{\nu(\beta_H, f, h, g)}{(\beta_H)^{hf}}. \tag{3.30}$$

Proof. The number of all state matrices representing IBFs with β cells, h hash functions and f inserted elements is given by (3.2). We apply Lemma 8 to obtain the number of state matrices which allow for extracting at least y elements. The inequality (3.30) follows by summing over all $y \leq g \leq f$. □

We note that the result of Lemma 8 gives us a lower bound on the number of state matrices which allow to extract at least g elements. For some state matrices in $(\mathbf{S}_{\beta_H,f})^h$, new elements may become available for extraction after g elements are removed from the initial IBF. Thus, the actual number of extracted elements is strictly larger or equal to the estimate. However, in Lemma 10, as we sum over all $y \leq g \leq f$, we already account for these additional elements.

On the other hand, the result in Lemma 8 is exact for $g = 0$ as the elements contained in the IBF are not changed. Thus

$$\Pr(Y_{\beta_H,h,f} = 0) = \frac{\zeta(\beta_H, f)^h}{(\beta_H)^{hf}}.$$

We can now state the complementary result to Theorem 2 which gives the lower bound for extraction success given an extraction rate.

Theorem 11. *An IBF with β cells, h hash functions and f inserted elements fails to achieve extraction rate r_{Extract} with probability less or equal than*

$$1 - \sum_{g=\lceil f r_{\mathsf{Extract}} \rceil}^{f} \frac{\nu(\beta_H, f, h, g)}{(\beta_H)^{hf}}.$$

Proof. In order to achieve the target extraction rate r_{Extract}, due to (2.4) we need to extract at least $y = \lceil f r_{\mathsf{Extract}} \rceil$ elements. As failing to extract at least y elements is a complementary event to succeeding to extract at least y elements, we can directly apply Lemma 10. □

3.1.3. Experimental results on partial IBF extraction

Before we look at the experimental results, we introduce another result for estimating the probability of failing to extract all elements from an IBF for comparison. The result is from [64] and given here as Theorem 12.

Theorem 12 ([64, Theorem 2])**.** *For given $\beta \geq 1$, $f \geq 1$ and $h \geq 1$, the probability of being able to extract all elements from an IBF is less or equal than*

$$\sum_{i=2}^{f} \binom{f}{i} \left(\frac{\zeta(\beta_H, i)}{(\beta_H)^i} \right)^h.$$

The previous results give the probability that the $\mathsf{Extract}()$ procedure fails to extract all of the inserted elements. However, using Theorem 11 we can compute the failure probability of the $\mathsf{Extract}()$ procedure returning only a subset of the inserted elements. This allows us to gain additional insight into IBF efficiency in edge cases.

As mentioned earlier, these result require that the IBFs are initialized with parameters which provide sufficient overhead. However, this requires prior knowledge on the number of elements being inserted into an IBF. As we later see in the construction of a data synchronization protocol in Section 3.2, this assumption does not hold and requires additional protocol on top of data synchronization protocol. Independently, the requirement for prior knowledge of element count prevents using IBFs in a setting where the elements to be inserted into an IBF are obtained from a stream.

For comparative results, we compute the probabilities for failing to achieve extraction rate $r_{\mathsf{Extract}} = 1$ using Theorems 2 and 12. For

$$r_{\mathsf{Extract}} \in \{0.1, 0.2, 0.5, 1\} \tag{3.31}$$

we use Theorem 11.

Additionally, for every choice of parameters, we compare the theoretical estimate to the outcome of a numerical simulation. For numerical simulations, for every run we uniformly sample f elements from a finite field $\mathbb{F}_p$ where p is a 256-bit prime. We initialized h hash functions $H_1, \ldots H_h$ by randomly sampling a 256-bit random seed seed_i, $i \in [h]$, and defining

$$H_i(x) \triangleq (i-1)\beta_H + (\text{SHA-256}(\text{seed}_i \mid x) \bmod \beta_H) + 1,$$

where $\text{seed}_i \mid x$ is a concatenation of bit-string representations of seed_i and x. Using the elements and hash functions, we initialized an IBF $\mathcal{F}$ with β cells and inserted all sampled elements to $\mathcal{F}$.

For every run, we extract the elements from $\mathcal{F}$, compute and store the extraction rate r_{Extract}. Finally, we count the number of runs where the actual extraction rate is above the parameters given in (3.31).

The number of runs for every set of parameters is 10000. The number of cells in an IBF β is fixed to 120, the number of hash functions is $h \in \{2,3,4,5\}$, and the number of inserted elements is $f \in \{20, 40, 60, 80, 100, 120\}$. The parameter choices for the number of hash functions is motivated by simulations from [22], where $h = 3$ and $h = 4$ outperformed other choices. We included $h = 2$ and $h = 5$ for additional insight.

The theoretical and simulation results are given in Tables 7, 8, 9 and 10.

In the tables, we denote that Theorem 2 fails if the number of inserted elements is above the threshold, i.e. $f > \beta\chi_h$ for χ_h as given in Table 2. Additionally, for $h = 2$, we used the value $\chi_2 = 2$. We denote that Theorem 12 fails if the result is larger or equal to 1.

We observe that the choices for the number of hash functions is critical for decreasing the probability of extraction failure. However, more specific behavior also depends on the overhead ratio of IBF cell count to the number of inserted elements.

If the number of inserted elements equals the number of cells in the IBF, i.e. overhead ratio of 1, then a smaller number of hash functions allows to partially extract more elements both in simulations and in theory using Theorem 11. The extraction almost always fails if $r_{\mathsf{Extract}} \in \{0.5, 1\}$ for all choices of h in simulation and in theory. The extraction also almost always fails if $h \in \{4,5\}$ and if $r_{\mathsf{Extract}} \in \{0.1, 0.2\}$. However, if $h = 2$ then it is always possible to extract at least $r_{\mathsf{Extract}} = 0.1$ elements and almost always possible to extract at least $r_{\mathsf{Extract}} = 0.2$ elements. If $h = 3$, then it with low failure probability it is possible to extract $r_{\mathsf{Extract}} = 0.1$ elements and high failure probability $r_{\mathsf{Extract}} = 0.2$ elements.

Table 7. IBF extraction failure probabilities for $\beta = 120$ and $h = 2$

f	Theorem 2	Theorem 12	r_{Extract}	Theorem 11	Simulation
20	$1.05 \cdot 10^{-1}$	$5.64 \cdot 10^{-2}$	0.1	$2.38 \cdot 10^{-16}$	0
			0.2	$3.48 \cdot 10^{-13}$	0
			0.5	$1.73 \cdot 10^{-6}$	0
			1	$7.40 \cdot 10^{-1}$	$2.89 \cdot 10^{-2}$
40	$4.30 \cdot 10^{-1}$	$3.07 \cdot 10^{-1}$	0.1	$1.99 \cdot 10^{-18}$	0
			0.2	$1.49 \cdot 10^{-13}$	0
			0.5	$1.58 \cdot 10^{-4}$	0
			1	1	$1.76 \cdot 10^{-1}$
60	$9.75 \cdot 10^{-1}$	fails	0.1	$3.68 \cdot 10^{-17}$	0
			0.2	$3.12 \cdot 10^{-11}$	0
			0.5	$3.86 \cdot 10^{-2}$	0
			1	1	$5.19 \cdot 10^{-1}$
80	fails	fails	0.1	$2.86 \cdot 10^{-14}$	0
			0.2	$6.56 \cdot 10^{-8}$	0
			0.5	$7.43 \cdot 10^{-1}$	$3.60 \cdot 10^{-3}$
			1	1	$9.40 \cdot 10^{-1}$
100	fails	fails	0.1	$1.04 \cdot 10^{-10}$	0
			0.2	$1.34 \cdot 10^{-4}$	0
			0.5	1	$2.82 \cdot 10^{-1}$
			1	1	1
120	fails	fails	0.1	$3.67 \cdot 10^{-7}$	0
			0.2	$4.36 \cdot 10^{-2}$	$7.00 \cdot 10^{-4}$
			0.5	1	$9.84 \cdot 10^{-1}$
			1	1	1

Table 8. IBF extraction failure probabilities for $\beta = 120$ and $h = 3$

f	Theorem 2	Theorem 12	r_{Extract}	Theorem 11	Simulation
20	$1.30 \cdot 10^{-2}$	$3.00 \cdot 10^{-3}$	0.1	$1.35 \cdot 10^{-19}$	0
			0.2	$5.11 \cdot 10^{-16}$	0
			0.5	$3.95 \cdot 10^{-8}$	0
			1	$6.58 \cdot 10^{-1}$	$5.00 \cdot 10^{-4}$
40	$5.35 \cdot 10^{-2}$	$1.28 \cdot 10^{-2}$	0.1	$1.73 \cdot 10^{-19}$	0
			0.2	$2.83 \cdot 10^{-14}$	0
			0.5	$1.51 \cdot 10^{-4}$	0
			1	1	$2.70 \cdot 10^{-3}$
60	$1.21 \cdot 10^{-1}$	$3.17 \cdot 10^{-2}$	0.1	$2.88 \cdot 10^{-15}$	0
			0.2	$1.99 \cdot 10^{-9}$	0
			0.5	$2.37 \cdot 10^{-1}$	0
			1	1	$7.90 \cdot 10^{-3}$
80	$2.17 \cdot 10^{-1}$	fails	0.1	$6.80 \cdot 10^{-10}$	0
			0.2	$1.65 \cdot 10^{-4}$	0
			0.5	$9.99 \cdot 10^{-1}$	$1.40 \cdot 10^{-3}$
			1	1	$3.35 \cdot 10^{-2}$
100	fails	fails	0.1	$3.77 \cdot 10^{-5}$	0
			0.2	$1.92 \cdot 10^{-1}$	$6.00 \cdot 10^{-4}$
			0.5	1	$5.50 \cdot 10^{-1}$
			1	1	$8.73 \cdot 10^{-1}$
120	fails	fails	0.1	$4.34 \cdot 10^{-2}$	$4.80 \cdot 10^{-3}$
			0.2	$9.80 \cdot 10^{-1}$	$3.89 \cdot 10^{-1}$
			0.5	1	1
			1	1	1

Table 9. IBF extraction failure probabilities for $\beta = 120$ and $h = 4$

f	Theorem 2	Theorem 12	r_{Extract}	Theorem 11	Simulation
20	$2.38 \cdot 10^{-3}$	$2.35 \cdot 10^{-4}$	0.1	$2.11 \cdot 10^{-3}$	0
			0.2	$2.11 \cdot 10^{-3}$	0
			0.5	$2.11 \cdot 10^{-3}$	0
			1	$6.36 \cdot 10^{-1}$	0
40	$9.77 \cdot 10^{-3}$	$9.74 \cdot 10^{-4}$	0.1	$1.84 \cdot 10^{-3}$	0
			0.2	$1.84 \cdot 10^{-3}$	0
			0.5	$2.72 \cdot 10^{-3}$	0
			1	1	$3.00 \cdot 10^{-4}$
60	$2.22 \cdot 10^{-2}$	$2.25 \cdot 10^{-3}$	0.1	$2.95 \cdot 10^{-3}$	0
			0.2	$2.96 \cdot 10^{-3}$	0
			0.5	$8.21 \cdot 10^{-1}$	0
			1	1	0
80	$3.96 \cdot 10^{-2}$	fails	0.1	$4.93 \cdot 10^{-3}$	0
			0.2	$9.33 \cdot 10^{-2}$	$1.00 \cdot 10^{-4}$
			0.5	1	$1.85 \cdot 10^{-2}$
			1	1	$2.48 \cdot 10^{-2}$
100	fails	fails	0.1	$1.03 \cdot 10^{-1}$	$8.60 \cdot 10^{-3}$
			0.2	$9.83 \cdot 10^{-1}$	$3.38 \cdot 10^{-1}$
			0.5	1	$9.93 \cdot 10^{-1}$
			1	1	$9.99 \cdot 10^{-1}$
120	fails	fails	0.1	$9.03 \cdot 10^{-1}$	$6.09 \cdot 10^{-1}$
			0.2	1	$9.98 \cdot 10^{-1}$
			0.5	1	1
			1	1	1

Table 10. IBF extraction failure probabilities for $\beta = 120$ and $h = 5$

f	Theorem 2	Theorem 12	r_{Extract}	Theorem 11	Simulation
20	$5.71 \cdot 10^{-4}$	$2.39 \cdot 10^{-5}$	0.1	$6.39 \cdot 10^{-3}$	0
			0.2	$6.39 \cdot 10^{-3}$	0
			0.5	$6.39 \cdot 10^{-3}$	0
			1	$6.56 \cdot 10^{-1}$	0
40	$2.34 \cdot 10^{-3}$	$9.83 \cdot 10^{-5}$	0.1	$8.52 \cdot 10^{-3}$	0
			0.2	$8.52 \cdot 10^{-3}$	0
			0.5	$1.80 \cdot 10^{-2}$	0
			1	1	0
60	$5.32 \cdot 10^{-3}$	$2.64 \cdot 10^{-4}$	0.1	$1.56 \cdot 10^{-2}$	0
			0.2	$1.68 \cdot 10^{-2}$	0
			0.5	$9.97 \cdot 10^{-1}$	0
			1	1	0
80	$9.50 \cdot 10^{-3}$	fails	0.1	$5.05 \cdot 10^{-2}$	$1.00 \cdot 10^{-3}$
			0.2	$8.24 \cdot 10^{-1}$	$5.06 \cdot 10^{-2}$
			0.5	1	$4.29 \cdot 10^{-1}$
			1	1	$4.45 \cdot 10^{-1}$
100	fails	fails	0.1	$8.47 \cdot 10^{-1}$	$4.94 \cdot 10^{-1}$
			0.2	1	$9.83 \cdot 10^{-1}$
			0.5	1	1
			1	1	1
120	fails	fails	0.1	1	$9.96 \cdot 10^{-1}$
			0.2	1	1
			0.5	1	1
			1	1	1

If the overhead ratio is larger than one but smaller than the threshold in Table 2, then the behavior is similar. The smallest extraction failure probability is obtained with $h = 2$ and is slightly larger for $h = 3$. However, we start seeing that there are a few cases in simulation where $h = 3$ allows extracting all elements when $h = 2$ does not. For $h \in \{4,5\}$ the failure probability of being able to extract even $r_{\mathsf{Extract}} = 0.1$ elements is high.

If the overhead ratio is approximately the threshold in Table 2, then with $h \in \{3,4\}$ the failure probability for extracting all elements is around 10^{-2} and $h \in \{2,5\}$ almost always fails. Between $h \in \{3,4\}$, $h = 3$ behaves better in case $r_{\mathsf{Extract}} < 1$.

For the overhead ratio above the threshold in Table 2, larger number of hash functions ensure that there are less extraction failures. We observed that for extracting all elements from an IBF, with $h = 5$ there were no failures, with $h = 4$ there is a negligible number of failures and with $h = 3$ the number of failures was low. However, for $h = 2$ the failure probability for being able to extract all elements is high.

In conclusion, even though if the overhead ratio is fixed, other h values are optimal, then $h = 3$ is the best choice for different cases while having close to optimal extraction failure probabilities.

3.2. Iterative data synchronization

3.2.1. Data synchronization using IBFs

Data synchronization using IBFs was first proposed in [28] and then studied in [22]. In this section, we remind the construction in [28] and consider an improvement using partially extractable IBFs as proposed in the previous section.

Let the network consist of two nodes A and B. The nodes have the corresponding sets $\mathcal{S}_A \subseteq \mathbb{F}$ and $\mathcal{S}_B \subseteq \mathbb{F}$. We denote the symmetric difference of the sets as

$$\mathcal{S}_\Delta \triangleq \mathcal{S}_A \triangle \mathcal{S}_B$$

and its cardinality as

$$d \triangleq |\mathcal{S}_\Delta| . \tag{3.32}$$

We additionally define a procedure $\mathsf{Add}()$ for adding two IBFs $\mathcal{F}_1$ and $\mathcal{F}_2$ which contains all elements inserted into $\mathcal{F}_1$ and $\mathcal{F}_2$. The description of the procedure $\mathsf{Add}()$ is given in Algorithm 6. Even though the current description is given for two input IBFs, we see that the description could trivially be generalized to any number of input IBFs.

By comparing the definitions of procedures $\mathsf{Insert}()$ and $\mathsf{Add}()$ (Algorithms 2 and 6), we can see that the IBF $\mathcal{F}$ returned by $\mathsf{Add}()$ procedure is constructed as if all the elements inserted into $\mathcal{F}_1$ and $\mathcal{F}_2$ were inserted to $\mathcal{F}$. Thus, all the results applying to an individual IBF $\mathcal{F}'$ apply to IBF which is obtained using $\mathsf{Add}()$ procedure.

Algorithm 6 Add two IBFs

```
1:  procedure Add(F_1, F_2)
2:      F ← Init(h, β, F, F')
3:      for all j ∈ [β] do
4:          c_j ← F[j]
5:          c̄_j ← F_1[j]
6:          ĉ_j ← F_2[j]
7:          c_j.count ← c̄_j.count + ĉ_j.count
8:          c_j.val ← c̄_j.val + ĉ_j.val
9:          c_j.ch ← c̄_j.ch + ĉ_j.ch
10:         F[j] ← c_j
11:     return F
```

For a two-node data synchronization protocol we assume that the domain of elements is $\mathbb{F}_{2^n}$, the checksum hash range is $\mathbb{F}_{2^\gamma}$ and the `count` field is defined over $\mathbb{F}_2$. In this case, the addition of the elements in the corresponding fields in an IBF cell corresponds to the bit-wise XOR operation.

A high-level idea of the data synchronization protocol between the two nodes is that the nodes construct IBFs $\mathcal{F}_A$ and $\mathcal{F}_B$ of their sets and exchange them. The exchanged IBFs are then added to the local IBFs, resulting in an IBF $\mathcal{F}$ which contain elements both from $\mathcal{S}_A$ and $\mathcal{S}_B$. Due to Lemma 1, the resulting IBF would allow to extract the elements in the union of the sets if the initial IBF are constructed with large enough β.

However, as we required that the IBFs are defined over a binary field, we obtain a significant performance improvement. As the elements which belong to the intersection $\mathcal{S}_A \cap \mathcal{S}_B$ appear both in $\mathcal{F}_A$ and $\mathcal{F}_B$, then during the addition they cancel out as $x + x = 0$ for any x in $\mathbb{F}_2$ and the corresponding extension fields $\mathbb{F}_{2^n}$ and $\mathbb{F}_{2^\gamma}$. Thus, the resulting $\mathcal{F}$ obtained by adding the IBFs contain only the elements in the symmetric set difference $\mathcal{S}_\Delta$.

Now, if the nodes A and B have an estimate $\widetilde{d}$ on the size of $\mathcal{S}_\Delta$, they can initialize their IBFs such that all elements can be extracted from $\mathcal{F}$. The protocol that summarizes the high-level description is given in Protocol 4.

We formally give the result on the correctness of the Protocol 4 as Lemma 13.

Lemma 13. *If $p = 2$ and $\widetilde{d} \geq d$, then pairwise data synchronization protocol given in Protocol 4 is correct with probability $O(\widetilde{d}^{-h+2})$.*

Proof. Follows directly from previous discussion and Theorem 2. □

In order for Protocol 4 to function properly, there needs to be an initial estimate on the size of symmetric difference. In [22], the authors describe a data structure called Strata Estimator. The idea of Strata Estimator is to split the domain $\mathbb{F}_{2^n}$ into n partitions $\mathbb{F}_{2^1}, \ldots, \mathbb{F}_{2^n}$ of increasing size and perform data synchronization between elements of the corresponding partitions. When encountering a partition

Protocol 4 Pairwise data synchronization

Label

Π_4

Network topology

Nodes $\mathcal{V} = \{A, B\}$, edges $\mathcal{E} = \{(A,B),(B,A)\}$.

Randomness model

Private randomness

Input

Node A possesses $\mathcal{S}_A$ and B possesses $\mathcal{S}_B$.

Goal

Both nodes obtain $\mathcal{S}_{\texttt{REC}} = \mathcal{S}_A \cup \mathcal{S}_B$.

Offline phase

Nodes have estimated the size $\widetilde{d}$ of symmetric set difference $\mathcal{S}_A \triangle \mathcal{S}_B$. Nodes have agreed on h and γ.

Online phase

$r = 1$

Computation phase

1. Node A initializes $\mathcal{F}_A \leftarrow \mathsf{Init}(h, \chi_h\widetilde{d}, \mathbb{F}_{2^n}, \mathbb{F}_{2^\gamma})$.
2. Node B initializes $\mathcal{F}_B \leftarrow \mathsf{Init}(h, \chi_h\widetilde{d}, \mathbb{F}_{2^n}, \mathbb{F}_{2^\gamma})$.
3. For every $x \in \mathcal{S}_A$ node A does $\mathsf{Insert}(\mathcal{F}_A, x)$.
4. For every $x \in \mathcal{S}_B$ node B does $\mathsf{Insert}(\mathcal{F}_B, x)$.

Transmission phase

5. Node A transmits $\mathcal{F}_A$ to node B.
6. Node B transmits $\mathcal{F}_B$ to node A.

Recovery phase

7. Nodes A and B compute $\mathcal{F}_\Delta \leftarrow \mathsf{Add}(\mathcal{F}_A, \mathcal{F}_B)$.
8. Nodes A and B extract $\mathcal{S}_\Delta \leftarrow \mathsf{Extract}(\mathcal{F}_\Delta)$.
9. Node A obtains $\mathcal{S}_{\texttt{REC}} \leftarrow \mathcal{S}_A \cup \mathcal{S}_\Delta$.
10. Node B obtains $\mathcal{S}_{\texttt{REC}} \leftarrow \mathcal{S}_B \cup \mathcal{S}_\Delta$.

where the protocol fails, the parties can estimate the actual difference size from the failing partition.

We describe the procedures $\mathsf{StrataEncode}()$ and $\mathsf{StrataDecode}()$ for constructing a Strata Estimator and using it for estimating the size difference in Algorithms 7 and 8.

The procedure $\mathsf{StrataEncode}()$ uses an hash function $H_{\text{strata}} : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ which ensures that the output is uniformly distributed in $\mathbb{F}_{2^n}$. By counting the number of trailing zeroes in the output, the procedure decides on an IBF where the element is inserted. As the number of trailing zeroes is unique for every element, then we obtain partitions of $\mathbb{F}_{2^n}$ of decreasing sizes. The procedure outputs the array of IBF for transfer.

Algorithm 7 Encode a Strata Estimator

procedure StrataEncode($\mathcal{S}$)
  for all $i \in [n]$ **do**
    $\mathcal{F}_i \leftarrow \mathsf{Init}(h, \beta, \mathbb{F}_{2^n}, \mathbb{F}_{2^\gamma})$
  for all $x \in \mathcal{S}$ **do**
    $i \leftarrow \mathsf{TrailingZeroes}(H_{\text{strata}}(x))$
    $\mathsf{Insert}(\mathcal{F}_i, x)$
  $\mathcal{Q} \leftarrow \langle \mathcal{F}_1, \ldots, \mathcal{F}_n \rangle$
  return $\mathcal{Q}$

We also note that the parameters for constructing the IBFs are defined on the protocol level. It was suggested in [22] to have $h = 4$ and $\beta = 80$.

Algorithm 8 Decode a Strata Estimator

procedure StrataDecode($\mathcal{Q}, \mathcal{S}$)
  $\texttt{count} \leftarrow 0$
  $\mathcal{Q}' \leftarrow \mathsf{StrataEncode}(\mathcal{S})$
  $i \leftarrow n$
  while $i \geq 0$ **do**
    $\mathcal{F}_i \leftarrow \mathsf{Add}(\mathcal{Q}[i], \mathcal{Q}'[i])$
    $\mathcal{S}_i \leftarrow \mathsf{Extract}(\mathcal{F}_i)$
    if $|\mathcal{F}_i| > 0$ **then**
      return $2^{i+1} \cdot \texttt{count}$
    $\texttt{count} \leftarrow \texttt{count} + |\mathcal{S}_i|$
    $i \leftarrow i - 1$
  return count

The StrataDecode() procedure receives the encoded strata and a local hasset. It starts looping over from IBF which corresponds to the smallest partition and constructs the IBF which corresponds to the symmetric set difference. If the decoding of IBF succeeds, then the number of extracted elements are added to the count and the procedure continues with an IBF corresponding to a larger partition. However, if complete extraction fails from the IBF, then the procedure assumes that the rest of IBFs contain the same proportion of elements and returns the approximation.

The following Theorem 14 states that for any given error rate, it is possible to construct a Strata Estimator which approximates the actual difference size. By combining it with Protocol 4, it is possible to obtain a complete pairwise data synchronization protocol.

Theorem 14 ([22, Theorem 2])**.** *Let ε and δ be constants in the interval $(0,1)$, an let $\mathcal{S}_A$ and $\mathcal{S}_B$ be two sets where* (3.32) *holds. If we encode the two sets with Strata Estimator, in which each IBF in the estimator has β cells using h hash functions, where β and h depend only on ε and δ, then with probability at least $1-\varepsilon$ it is possible to estimate d within a factor of $1 \pm \delta$.*

3.2.2. Iterative data synchronization

We propose an alternative approach for data synchronization which allows for underestimating the size of symmetric difference d. For example, in order to have a compact Strata Estimator, the parameters in Theorem 14 have to be chosen carefully to balance between the estimator size and its error rate. If the parameters are chosen inadequately, the Strata Estimator outputs an underestimation and Protocol 4 fails. With the use of a protocol which allows underestimation, it is still possible to recover and perform full data synchronization.

For the description of the improved protocol, we need an additional hash function H_{set} which is used for testing set equality. It takes as input a subset of $\mathbb{F}_{2^n}$ and outputs a short value to ensure low collision rate, e.g. in $\mathbb{F}_{2^{256}}$. For defining the hash functions H_i, $i \in [h]$, we assume that there exists well-defined hash function H and define

$$H_i(x) \triangleq (i-1)\beta_H + (H(\text{seed}_i \mid x) \bmod \beta_H) + 1, \tag{3.33}$$

using randomly sampled seed seed_i. The description of H_i is then given as (seed_i, i).

The full description for iterative data synchronization protocol between two parties is given in Protocol 5. We show in Lemma 15 that with probability close to 1, the protocol succeeds to synchronize the sets for any estimate $\widetilde{d}$.

The following Lemma 15 shows that for any estimate $\widetilde{d}$ the protocol is correct.

Lemma 15. *For any initial has-sets $\mathcal{S}_A^{(1)}$ and $\mathcal{S}_B^{(1)}$, Protocol 5 terminates with high probability with both A and B possessing $\mathcal{S}_A^{(1)} \cup \mathcal{S}_B^{(1)}$ if $\beta_H > 1$.*

Proof. Similarly to the discussion in proof of Lemma 13, we observe that the IBF $\mathcal{F}_\Delta^{(r)}$ corresponds to the set $\mathcal{S}_A^{(r)} \triangle \mathcal{S}_B^{(r)}$ and thus the extracted elements $\mathcal{S}_\Delta^{(r)}$ is a subset of $\mathcal{S}_A^{(r)} \triangle \mathcal{S}_B^{(r)}$.

If in a round r we have $\left|\mathcal{S}_\Delta^{(r)}\right| > 0$, then

$$\mathcal{S}_A^{(r+1)} \triangle \mathcal{S}_B^{(r+1)} \subsetneq \mathcal{S}_A^{(r)} \triangle \mathcal{S}_B^{(r)}$$

and after sufficient number of rounds there exists r' such that

$$\mathcal{S}_A^{(r')} \triangle \mathcal{S}_B^{(r')}$$

which corresponds to $\mathcal{S}_A^{(r')} = \mathcal{S}_B^{(r')}$.

We now ensure that the probability that $\left|\mathcal{S}_\Delta^{(r)}\right| > 0$ is non-zero. This holds due to Lemma 10 by taking $y = 1$ as $\nu(\beta_H, f, h, 1) > 0$ if $\beta_H > 1$. □

3.2.3. Experimental results

As the number of rounds in Protocol 5 is not deterministic and the parties halt the protocol only if the corresponding set-hashes $H_{\text{set}}(\mathcal{S}_A^{(r)})$ and $H_{\text{set}}(\mathcal{S}_B^{(r)})$ are equal in

Protocol 5 Iterative pairwise data synchronization

Label

Π_5

Network topology

Nodes $\mathcal{V} = \{A, B\}$, edges $\mathcal{E} = \{(A,B),(B,A)\}$.

Randomness model

Private randomness

Input

Nodes A and B have initial has-sets $\mathcal{S}_A^{(1)}$ and $\mathcal{S}_B^{(1)}$.

Goal

Both nodes obtain $\mathcal{S}_{\mathtt{REC}} = \mathcal{S}_A^{(1)} \cup \mathcal{S}_B^{(1)}$.

Offline phase

Nodes have agreed on $\widetilde{d}$, h and γ.

Online phase

For $r \in \mathbb{N}$

Computation phase

1. Node A computes $\mathtt{ch}_A^{(r)} = H_{\text{set}}(\mathcal{S}_A^{(r)})$.
2. Node B computes $\mathtt{ch}_B^{(r)} = H_{\text{set}}(\mathcal{S}_B^{(r)})$.
3. Node A samples $\text{seed}_i^{(r)}$ for $i \in [h]$.
4. Node A initializes $\mathcal{F}_A^{(r)} \leftarrow \mathsf{Init}(h, \chi_h \widetilde{d}, \mathbb{F}_{2^n}, \mathbb{F}_{2^\gamma})$ using hash functions as in (3.33) with $\text{seed}_i^{(r)}, i \in [h]$.
5. For every $x \in \mathcal{S}_A^{(r)}$, node A does $\mathsf{Insert}(\mathcal{F}_A^{(r)}, x)$.

Transmission phase

6. Node A transmits $\mathtt{ch}_A^{(r)}$ to B.
7. Node B checks that $\mathtt{ch}_A^{(r)} \neq \mathtt{ch}_B^{(r)}$ and aborts otherwise.
8. Node A transmits $\text{seed}_i^{(r)}$, $i \in [h]$.
9. Node A transmits $\mathcal{F}_A^{(r)}$.

Recovery phase

10. Node B initializes $\mathcal{F}_B^{(r)}$ using hash functions with received $\text{seed}_i^{(r)}$, $i \in [h]$.
11. For every $x \in \mathcal{S}_B^{(r)}$, node B does $\mathsf{Insert}(\mathcal{F}_A^{(r)}, x)$.
12. Node B computes $\mathcal{F}_\Delta^{(r)} \leftarrow \mathsf{Add}(\mathcal{F}_A^{(r)}, \mathcal{F}_B^{(r)})$.
13. Node B extracts $\mathcal{S}_\Delta^{(r)} \leftarrow \mathsf{Extract}(\mathcal{F}_\Delta^{(r)})$.
14. Node B takes $\mathcal{S}_B^{(r+1)} \leftarrow \mathcal{S}_B^{(r)} \cup \mathcal{S}_\Delta^{(r)}$.
15. Nodes A and B swap sides, take $r \leftarrow r+1$ and go to step 1

Step 7, then it would be beneficial to be able to estimate the approximate number of rounds in the protocol. In this section, we apply the analysis of absorbing Markov chains for estimating the number of protocols rounds. Additionally, we compare the estimated values against the simulated runs.

Define the sequence of the random variables $R(i)$, $i \geq 0$, whose values denote the number of the elements in the symmetric difference $\mathcal{S}_A \triangle \mathcal{S}_B$ after completion of round i in Protocol 5. In the sequel, we call the variable $R(i)$ *the i-th state* of the protocol. We remark that the transition from the state $R(i)$ into the next state $R(i+1)$ does not depend on the states $R(j)$ for $j < i$, thus forming a Markov chain. It follows from Lemma 15 that for any its realization, the sequence $R(i)$ is monotonically non-increasing with i, and it approaches 0 for $i \to \infty$.

Next, we use Lemma 8 and the counting argument from the proof of Theorem 2 for estimation of the transition probabilities between different states. If $R(i) \geq \beta/\chi_h$, then from Lemma 8 we have:

$$\Pr(R(i+1) = f - g \mid R(i) = f) = \frac{\nu(\beta_H, f, h, g)}{\beta_H^{hf}} \,. \tag{3.34}$$

For $R(i) < \beta/\chi_h$ it was observed in [28] and [64] that the extraction failure probability was dominated by the case where two elements were inserted in the same IBF cells. From (2.5), this yields the probability that two elements are left in the IBF after extraction as

$$\Pr\left(R(i+1) = 2 \mid R(i) < \frac{\beta}{\chi_h}\right) = \binom{f}{2}\binom{\beta}{h}\left(\frac{h}{\beta}\right)^{2h}, \tag{3.35}$$

and the probability that all elements are successfully extracted is

$$\Pr\left(R(i+1) = 0 \mid R(i) < \frac{\beta}{\chi_h}\right) = 1 - \binom{f}{2}\binom{\beta}{h}\left(\frac{h}{\beta}\right)^{2h}. \tag{3.36}$$

We observe that the result of Lemma 8 yields a lower bound on the number of extracted elements. Thus, the expected number of steps obtained from the relations (3.34), (3.35) and (3.36) imply an upper bound on the expected number of rounds in Protocol 5.

We compute the expected number of steps for $\beta = 120$, $h \in \{2,3,4,5\}$ and $f \in [20,200]$. For $h = 2$, we used $\chi_2 = 2$. For $h > 2$, the values of χ_h are as in Table 2. The results are shown in Figures 10 for the case $f < \beta/\chi_h$ (where relations (3.35) and (3.36) apply) and Figure 11 for the case $f \geq \beta/\chi_h$ (where relation (3.34) applies).

We compare the numerical bounds with the simulation results, for the same choices of β, h and $f \in [20,200]$. For every set of parameters, we ran the protocol 100 times with randomly chosen elements and hash functions. The elements are chosen from $\mathbb{F}$ uniformly at random, one by one, while ensuring that the same

element is not chosen twice. The domain $\mathbb{F}$ is the field of 256-bit long integers modulo the prime number

$$p = 115792089237316195423570985008687907853269984665640564039457584007913129640233.$$

For the hash functions, we use the 256-bit long version of SHA-2 with uniformly chosen 32-bit long random seeds [53]. The received value is converted into an integer, and its residue modulo β_H is used as the index of the cell in the subfilter. The average number of rounds for full data synchronization are shown in Figures 10 and 11 by using solid lines.

In Figure 10, we can see that for some cases, the number of rounds for full data synchronization exceeds the theoretical bounds. This is due to the claim in the proof of Theorem 2 that the failure probability of extracting all elements is dominated by the case where two elements remain in the IBF. This claim in general holds for the case $\beta \to \infty$. For fixed values of β, the probability of cases of having three- or more elements collide in the IBF is non-negligible and the bound given by (3.35)-(3.36) does not take these cases into account.

We observe in Figure 11 that the analytical estimates are quite close to the simulated results. We also observe that, for the selected parameters, the number of rounds is smaller for $h \in \{3,4\}$ than for $h = 5$ in both under- and over-threshold IBFs. By comparing the results for $h = 3$ and $h = 4$, we observe that the performance is similar for the under-threshold IBF, but the choice $h = 3$ allows for a smaller number of rounds for over-threshold IBF. Since the threshold for $h = 3$ is larger than for $h = 4$, larger number of elements can be synchronized in a single round. We also observe that for $h = 2$, the protocol underperforms when IBF is under-threshold. In this case the performance is weaker than for $h = 3$ if $f \leq \beta$, yet it allows for a smaller number of rounds in the case where $f > \beta$.

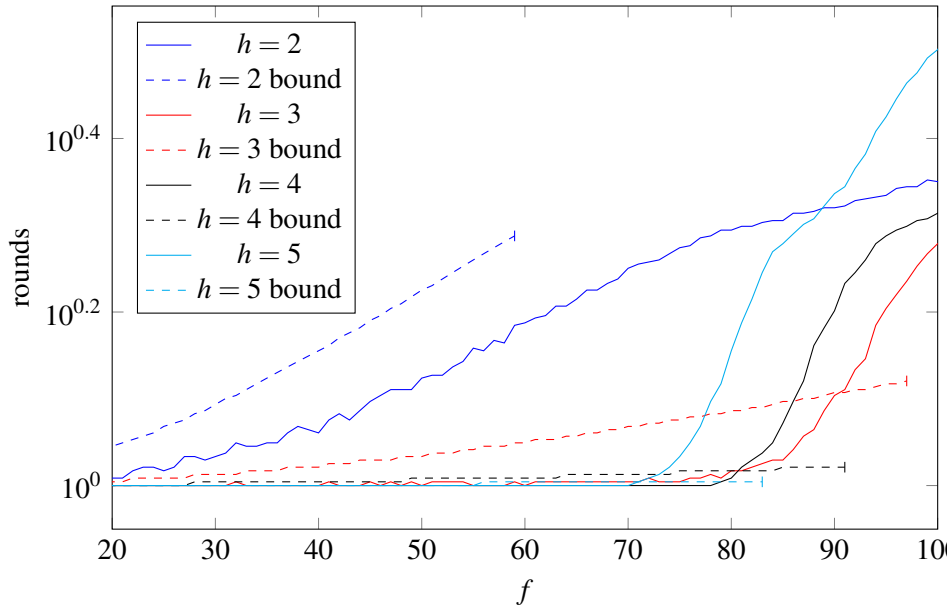

Figure 10. Comparison of experimental and theoretical number of rounds for iterative data synchronization when $f < \beta/\chi$. Relations (3.35) and (3.36) are used to compute the upper bound.

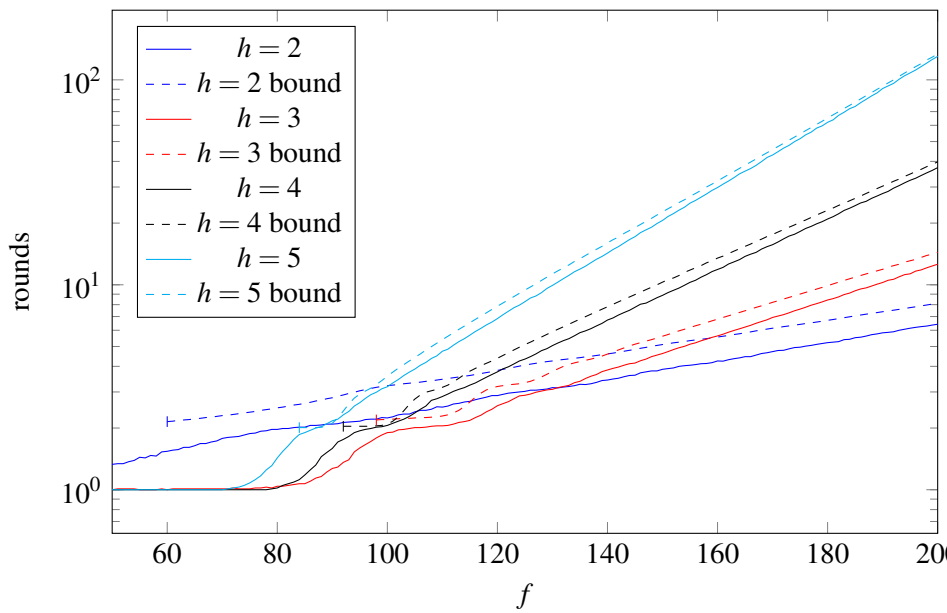

Figure 11. Comparison of experimental and theoretical number of rounds for iterative data synchronization when $f \geq \beta/\chi$. Relation (3.34) is used to compute the upper bound.

4. DATA DISSEMINATION PROBLEM

In this chapter, we generalize the index coding and data exchange problem defined in Sections 2.8.1 and 2.8.2. For that, we first give a definition of ρ-solvable network in Section 4.1.

In Section 4.1.1 we consider 1-solvable network topologies and give a complete description of an optimal protocol which satisfies the requests of all nodes in a single round. The main result is given as Theorem 18.

We extend the method to any ρ-solvable network topology in Section 4.1.2. In order to solve the data distribution problem in arbitrary networks, we present a new algebra which allows to apply matrix operations to study the data dissemination in the network over many rounds. The main result is given as Theorem 24.

We conclude the chapter with the simulation results in Section 4.2.

4.1. Data distribution problem in an arbitrary network

In this section, we generalize and extend the results in Sections 2.8.1 and 2.8.2 to different types of network topologies.

To recall the setting from Section 2.8, there is an information vector $\mathbf{x} = (x_1, \ldots, x_k) \in \mathbb{F}^k$ of length k. Every node already knows some elements of the information vector. For every node $v_i \in \mathcal{V}$, the indices of the known elements are denoted as $\mathcal{Z}_{v_i} \subseteq [k]$. Every node also requests some of the elements of $\mathbf{x}$ it is missing. The indices of requested elements are denoted as $\mathcal{T}_{v_i} \subseteq [k]$.

We assume that the transmissions are broadcast and the protocols are deterministic. As described in Section 2.3.1, we assume in general that the transmissions are performed in rounds and we denote the number of rounds in the protocol as ρ.

When considering different network topologies and assignment of has- and request-sets to the nodes, the specific data distribution problem instance may not always allow to construct a protocol where every node can recover the elements in the request-set. In order to ensure that the instance can be solved, we introduce the following definition which allows us to consider the feasibility of the data distribution problem instance.

Definition 5. Consider a network topology based on the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The assignment of the sets $\mathcal{Z}_{v_i}$ and $\mathcal{T}_{v_i}$ is called *feasible* if for any $j \in \mathcal{T}_{v_i}$, $v_i \in \mathcal{V}$, there exists a node $v_\ell \in \mathcal{V}$ with $j \in \mathcal{Z}_{v_\ell}$ such that there is a finite directed path from ℓ to i in $\mathcal{G}$.

Definition 5 describes the condition for existence of a protocol, in which the nodes recover the requested elements. This can be achieved by transmitting the un-encoded elements from every node until all requests are eventually satisfied. Next, we define a measure on the minimal number of rounds in any protocol for a feasible instance of data distribution problem.

Definition 6. The network topology based on the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is said to be ρ-solvable, $\rho \in \mathbb{N}$, if for any feasible assignment of the sets $\mathcal{Z}_{v_i}$ and $\mathcal{T}_{v_i}$, $v_i \in \mathcal{V}$, ρ communication rounds are sufficient for the protocol to satisfy all the node requests, but $\rho - 1$ rounds are not sufficient. If the network is not ρ-solvable for any $\rho \in \mathbb{N}$, then we say that it is not solvable.

The following simple Lemma 16 defines the condition for a data distribution problem to be ρ-solvable.

Lemma 16. *The network topology based on graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is ρ-solvable for some $\rho \in \mathbb{N}$ if the maximum of the shortest length directed path from the node v_i to the node v_j for any two nodes $v_i, v_j \in \mathcal{V}$ is exactly ρ.*

Proof. Let there be a network topology $\mathcal{G}$ such that ρ is the maximum of the shortest length between any nodes $v_i, v_j \in \mathcal{V}$. Then, if all nodes transmit all elements in their current has-sets in every round, all nodes recover all elements in at least ρ rounds. This ensures that every node also recovers all requested elements.

On the other hand, let v_i and v_j be specific nodes where the shortest path between them has length ρ. Define $\mathcal{Z}_{v_i} = \{1\}$ and $\mathcal{Z}_{v_\ell} = \varnothing$ for $v_\ell \in \mathcal{V} \setminus \{v_i\}$. Define $\mathcal{T}_{v_j} = \{1\}$ and $\mathcal{T}_{v_\ell} = \varnothing$ for all $v_\ell \in \mathcal{V} \setminus \{v_j\}$. Then, for any protocol we require at least ρ rounds to satisfy request of node v_j. □

Next, we also define matrix representation of a graph which allows us to apply algebraic methods to compute the ρ-solvability for ρ.

Definition 7. Let the number of nodes in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be $u = |\mathcal{V}|$. The transposed $u \times u$-dimensional integer matrix $\mathbf{D}$ is called *adjacency matrix* of a graph $\mathcal{G}$ if

$$(\mathbf{D})_{i,j} = \begin{cases} 1 & \text{if } (v_j, v_i) \in \mathcal{E} \\ 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}.$$

Ones on the main diagonal indicate that the nodes retain their memory over rounds.

Corollary 17. *The network topology based on the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is ρ-solvable if ρ is the smallest integer, $\rho > 0$, such that all the entries in the matrix $\mathbf{D}^\rho$ are strictly positive.*

Proof. The network topology corresponding to a graph $\mathcal{G}$ which is ρ-solvable has diameter ρ. It is well-known result that the lowest power of the corresponding adjacency matrix where all elements are non-zero, the exponent yields the diameter of the graph [18]. □

4.1.1. 1-solvable networks

In this section, we consider the data distribution problem instances, where the underlying network is 1-solvable.

Similarly to Section 2.8.2, we define a set $\mathbf{P}_{v_i} \subseteq \mathbf{S}^{\mathbb{F}}_{k\times k}$ of $k \times k$-dimensional matrices for every node $v_i \in \mathcal{V}$:

$$\mathbf{P}_{v_i} \triangleq \{\mathbf{Y}_{v_i} \in \mathbf{S}^{\mathbb{F}}_{k\times k} : \forall \ell \in [k], (\mathbf{Y}_{v_i})_{\ell,j} = 0 \text{ if } j \notin \mathcal{Z}_{v_i}\}. \tag{4.1}$$

Similarly, we define the set of matrices $\mathbf{P} \subseteq \mathbf{S}^{\mathbb{F}}_{uk\times k}$ which is a set of matrices consisting of concatenation of matrices in $\mathbf{P}_{v_i}$ for every v_i:

$$\mathbf{P} \triangleq \left\{ \begin{bmatrix} \mathbf{Y}_{v_1} \\ \vdots \\ \mathbf{Y}_{v_u} \end{bmatrix} : \mathbf{Y}_{v_i} \in \mathbf{P}_{v_i} \right\}. \tag{4.2}$$

Given the in-neighbors $\mathcal{W}_{\mathtt{in}(v_i)} = \{v_1, \ldots, v_\eta\}$ of the node v_i and a matrix $\mathbf{Y} \in \mathbf{P}$, we denote by $\mathbf{Y}_{\mathcal{W}_{\mathtt{in}(v_i)}}$ the matrix

$$\mathbf{Y}_{\mathcal{W}_{\mathtt{in}(v_i)}} \triangleq \begin{bmatrix} \mathbf{Y}_{v_1} \\ \vdots \\ \mathbf{Y}_{v_\eta} \end{bmatrix}.$$

For every node $v_i \in \mathcal{V}$ we define the $k \times k$-dimensional information matrix $\mathbf{Q}_{v_i}$ where

$$(\mathbf{Q}_{v_i})_{\ell,j} = \begin{cases} 1 & \text{if } \ell = j \text{ and } j \in \mathcal{Z}_{v_i} \\ 0 & \text{otherwise} \end{cases}. \tag{4.3}$$

Similarly, for every node v_i we define $k \times k$-dimensional query matrix $\mathbf{T}_{v_i}$ where

$$(\mathbf{T}_{v_i})_{\ell,j} = \begin{cases} 1 & \text{if } \ell = j \text{ and } j \in \mathcal{T}_{v_i} \\ 0 & \text{otherwise} \end{cases}. \tag{4.4}$$

We can now present the first result which generalizes the result of Theorem 5 to any network topology which is 1-solvable.

Theorem 18. *Let the network topology based on graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be 1-solvable. Let $\mathbf{P}$ be a set of $uk \times k$-dimensional matrices as defined in* (4.2)*. For every node $v_i \in \mathcal{V}$ the matrices $\mathbf{Q}_{v_i}$ and $\mathbf{T}_{v_i}$ are defined as in* (4.3) *and* (4.4)*, respectively. There exists a protocol with a single round to satisfy the requests of all nodes $\mathcal{V}$ using π packets, where*

$$\pi = \min_{\mathbf{Y} \in \mathbf{P}} \sum_{v_i \in \mathcal{V}} \operatorname{rank}(\mathbf{Y}_{v_i}), \tag{4.5}$$

subject to following constraint for every $v_i \in \mathcal{V}$:

$$\operatorname{rowspace}\left(\begin{bmatrix} \mathbf{Y}_{\mathcal{W}_{\mathtt{in}(v_i)}} \\ \mathbf{Q}_{v_i} \end{bmatrix} \right) \supseteq \operatorname{rowspace}(\mathbf{T}_{v_i}). \tag{4.6}$$

Additionally, such a protocol has the minimal number of transmitted packets over all linear protocols satisfying the node demands. If there is no such matrix $\mathbf{Y} \in \mathbf{P}$ which satisfies the constraints (4.6)*, then there is no linear protocol which satisfies the demands of the nodes' in a single round.*

Before we prove Theorem 18, we first state two technical Lemmas 19 and 20.

Lemma 19. *Let V be an ambient vector space and $W \subseteq V$ be a linear subspace. If there is a vector $\mathbf{x} \in V$ such that $\mathbf{x} \notin W$, then there exists $\mathbf{y} \in W^\top$ such that $\mathbf{x} \cdot \mathbf{y} \neq 0$.*

Proof. Assume that there exists $\mathbf{x} \in V$ such that $\mathbf{x} \notin W$. Suppose that for all $\mathbf{y} \in W^\top$ we have $\mathbf{x} \cdot \mathbf{y} = 0$. This implies that $\mathbf{x}$ belongs to the dual space of $W^\top$. However, as the dual space of a dual space is the space itself, then this implies that $\mathbf{x} \in W$ which contradicts the assumption. Thus, there must exist $\mathbf{y} \in W^\top$ such that $\mathbf{x} \cdot \mathbf{y} \neq 0$. □

Lemma 20. *Let V be an ambient vector space and $W \subseteq V$ be a linear subspace. If $\mathbf{x} \in W^\top$, then for every subspace $U \subseteq W$ and for every vector $\mathbf{y} \in U$ we have $\mathbf{x} \cdot \mathbf{y} = 0$.*

Proof. Let $\mathbf{x} \in W^\top$. By picking any $\mathbf{y} \in U$ we have that $\mathbf{y} \in W$ as $U \subseteq W$. Thus, by the definition of the dual space we have $\mathbf{x} \cdot \mathbf{y} = 0$. □

Proof of Theorem 18. We prove the claim in two steps. First, we show that if there exists a matrix $\mathbf{Y} \in \mathbf{P}$ such that the conditions (4.6) hold, then we can construct a protocol which allows to satisfy all the requests of the nodes.

In the second step, we show that for any protocol which satisfies the requests of the nodes, we can construct a corresponding matrix $\mathbf{Y} \in \mathbf{P}$ which satisfies the constraints (4.6). This also implies that if there is no such matrix $\mathbf{Y} \in \mathbf{P}$, then there is no one-round protocol.

1. Existence of the protocol:
   We define the encoding and decoding functions for every node $v_i \in \mathcal{V}$ such that every node can recover the requested elements described by the indices $\mathcal{T}_{v_i}$.
   Let $\mathbf{Y} \in \mathbf{P}$ be the matrix which minimizes (4.5) such that (4.6) holds for all $v_i \in \mathcal{V}$. We can assume that for every partition $\mathbf{Y}_{v_i}$ the first $\operatorname{rank}(\mathbf{Y}_{v_i})$ rows span the whole $\operatorname{rowspace}(\mathbf{Y}_{v_i})$. Denote the rank of $\mathbf{Y}_{v_i}$ as π_{v_i}.
   As (4.6) holds, this means that for every $t \in \mathcal{T}_{v_i}$ by (4.3) and (4.4) we have that there is a vector $\mathbf{e}_t$ as a row in $\mathbf{T}_{v_i}$ and it can be written as a linear combination of rows in $\mathbf{Y}_{\mathcal{W}_{\mathtt{in}(v_i)}}$ and $\mathbf{Q}_{v_i}$:

$$\mathbf{e}_t = \sum_{v_\ell \in \mathcal{W}_{\mathtt{in}(v_i)}} \sum_{j \in [\pi_{v_\ell}]} \lambda_{v_\ell, j} \mathbf{Y}_{v_\ell}^{[j]} + \sum_{j \in [k]} \alpha_j \mathbf{Q}_{v_i}^{[j]}, \tag{4.7}$$

   where $\lambda_{v_\ell, j}$ and α_j are elements in $\mathbb{F}$.
   Given $\mathbf{e}_t$, we can now recover x_t by $\mathbf{e}_t \cdot \mathbf{x} = x_t$. By replacing into (4.7), we get

$$x_t = \sum_{v_\ell \in \mathcal{W}_{\mathtt{in}(v_i)}} \sum_{j \in [\pi_{v_\ell}]} \lambda_{v_\ell, j} (\mathbf{Y}_{v_\ell}^{[j]} \cdot \mathbf{x}) + \sum_{j \in [k]} \alpha_j (\mathbf{Q}_{v_i}^{[j]} \cdot \mathbf{x}), \tag{4.8}$$

   Then, every node v_ℓ transmits the packets

$$(\mathbf{Y}_{v_\ell}^{[j]} \cdot \mathbf{x})_{j \in [\pi_{v_\ell}]} \tag{4.9}$$

   to all out-neighbors. Due to (4.1), the packets are computed using the elements the node v_ℓ possesses and thus is correctly defined.
   Every node v_i can compute $(\mathbf{Q}_{v_i}^{[j]} \cdot \mathbf{x})$ using the elements it possess and thus it can recover every requested element x_t for $t \in \mathcal{T}_{v_i}$ using (4.8).
   As every node v_i sends $\pi_{v_\ell} = \operatorname{rank}(\mathbf{Y}_{v_i})$ number of packets,

$$\pi = \sum_{v_i \in \mathcal{V}} \pi_{v_\ell} = \sum_{v_i \in \mathcal{V}} \operatorname{rank}(\mathbf{Y}_{v_i})$$

   gives the total number of sent packets over all nodes.
2. Minimality of the number of transmitted packets:
   Assume, that there exists an optimal linear protocol satisfying the requests of the nodes where the total number of packets which are sent is $\pi^{\mathtt{opt}}$.

Because we consider linear protocols, then this means that there exists a $\pi_{v_i}^{\mathtt{opt}} \times k$-dimensional matrix $\mathbf{Y}_{v_i}$ for every node v_i such that the node transmits the packets

$$(\mathbf{Y}_{v_i}^{[j]} \cdot \mathbf{x})_{j \in [\pi_{v_i}^{\mathtt{opt}}]}.$$

As every node transmits $\pi_{v_i}^{\mathtt{opt}}$ packets, the total number of transmitted packets can be written as

$$\pi^{\mathtt{opt}} = \sum_{v_i \in \mathcal{V}} \pi_{v_i}^{\mathtt{opt}}.$$

Next, we show that for all nodes v_i, if the node has requested for an element x_t, i.e. $t \in \mathbf{T}_{v_i}$, then the unit vector $\mathbf{e}_t$ belongs to the vector space $W_{v_i} \subseteq \mathbb{F}^k$, where

$$W_{v_i} \triangleq \sum_{v_\ell \in \mathcal{W}_{\mathtt{in}(v_i)}} \text{rowspace}(\mathbf{Y}_{v_\ell}) + \text{rowspace}(\mathbf{Q}_{v_i}). \tag{4.10}$$

For that, we fix a node $v_i \in \mathcal{V}$ and $t \in \mathcal{T}_{v_i}$. In contrary, lets assume that $\mathbf{e}_t \notin W_{v_i}$. Due to Lemma 19, there must exist some information vector $\mathbf{x}$ such that

$$\mathbf{e}_t \cdot \mathbf{x} \neq 0. \tag{4.11}$$

From (4.10) and Lemma 20 we have that $\mathbf{x} \cdot \mathbf{Y}_{v_\ell}^{[j]} = 0$ for every $v_\ell \in \mathcal{W}_{\mathtt{in}(v_i)}$ and $j \in [\pi_{v_\ell}^{\mathtt{opt}}]$ as $\text{rowspace}(\mathbf{Y}_{v_\ell}) \subseteq W_{v_i}$. Similarly, due to (4.3) we have $\mathbf{x} \cdot \mathbf{e}_j = 0$ for all $j \in \mathcal{Z}_{v_i}$. Rephrasing, this means that

- every node $v_\ell \in \mathcal{W}_{\mathtt{in}(v_i)}$ transmits only zeroes;
- the side information available to node v_i is only zeroes.

The only vector which satisfies these requirements is $\mathbf{0}$. However, due to (4.11), we had that $\mathbf{x}$ had nonzero element at location t and thus $\mathbf{x} \neq \mathbf{0}$. Thus we have a contradiction and $\mathbf{e}_t \in W_{v_i}$.
Now, we construct a matrix $\mathbf{Y}_{v_i}$ which belongs to the matrix set $\mathbf{P}_{v_i}$ for every node v_i. We define

$$\mathbf{Y}_{v_i}^{[j]} = \begin{cases} \mathbf{Y}_{v_i}^{[j]} & \text{if } j \in [\pi_{v_\ell}^{\mathtt{opt}}] \\ \mathbf{0} & \text{otherwise} \end{cases}. \tag{4.12}$$

As we did not add any vectors to the $\text{rowspace}(\mathbf{Y}_{v_i})$, then $\text{rowspace}(\mathbf{Y}_{v_i}) = \text{rowspace}(\mathbf{Y}_{v_i})$ and we have

$$W_{v_i} \triangleq \sum_{v_\ell \in \mathcal{W}_{\mathtt{in}(v_i)}} \text{rowspace}(\mathbf{Y}_{v_\ell}) + \text{rowspace}(\mathbf{Q}_{v_i})$$

As

$$\sum_{v_\ell \in \mathcal{W}_{\mathtt{in}(v_i)}} \text{rowspace}(\mathbf{Y}_{v_\ell}) + \text{rowspace}(\mathbf{Q}_{v_i}) = \text{rowspace}\left(\begin{bmatrix} \mathbf{Y}_{\mathcal{W}_{\mathtt{in}(v_i)}} \\ \mathbf{Q}_{v_i} \end{bmatrix} \right),$$

we obtain that (4.6) holds.
Finally, due to definition of $\mathbf{Y}_{v_i}$ in (4.12), we have that $\operatorname{rank}(\mathbf{Y}_{v_i}) \leq \pi_{v_i}^{\mathtt{opt}}$ for every $v_i \in \mathcal{V}$. Thus also the total number of transmissions is subject to

$$\sum_{v_i \in \mathcal{V}} \operatorname{rank}(\mathbf{Y}_{v_i}) \leq \pi^{\mathtt{opt}}.$$

As we assumed that the number of transmission in the protocol is optimal, we get

$$\sum_{v_i \in \mathcal{V}} \operatorname{rank}(\mathbf{Y}_{v_i}) = \pi^{\mathtt{opt}}.$$

Thus, also (4.5) also holds.

□

Theorem 18 describes only the communication complexity of data distribution problem in 1-solvable networks. The protocol which achieves the minimal number of transmitted bits is obtained by solving a matrix rank minimization problem which is on its own is an NP-hard problem [23]. However, if the set of matrices $\mathbf{P}$ is restricted, then faster methods exist for solving the minimization problem [4]. Alternatively, an approximate solution to matrix rank minimization can be found efficiently [46].

Example 12. Trivially, both a star and a complete graph are 1-solvable. This allows to use Theorem 18 for constructing protocols for both index coding and data exchange problem instances. ■

To conclude this section, we give the complete protocol for data distribution problem in 1-solvable networks. The protocol description is given in Protocol 6.

We emphasize that the computation complexity of Protocol 6 is dominated by the offline phase of determining the encoding and decoding functions. In the online phase, every node v only computes π_v inner products to encode and $\pi_v + k$ multiplications to decode the packets.

Example 13. Consider an example network in Figure 12. There are five nodes v_1, v_2, v_3, v_4 and v_5, which in total possesses three bits of information x_1, x_2 and x_3. We see that for any *feasible* assignment of has- and request-sets the requests could be satisfied in a single round, i.e. the network topology is 1-solvable.

The has-sets of the nodes are given through the corresponding indices of the information vector, i.e.

$$\mathcal{Z}_{v_1} = \{1,2\}, \quad \mathcal{Z}_{v_2} = \{2,3\}, \quad \mathcal{Z}_{v_3} = \{1\}, \quad \mathcal{Z}_{v_4} = \{2\}, \quad \mathcal{Z}_{v_5} = \{1,3\}.$$

The request-sets are also given through the indices of the information vector, i.e.

$$\mathcal{T}_{v_3} = \{2,3\}, \quad \mathcal{T}_{v_4} = \{1,3\}, \quad \mathcal{T}_{v_5} = \{2\}.$$

Protocol 6 Data distribution in 1-solvable networks

Label

Π_6

Network topology

1-solvable network with u nodes $\mathcal{V}$.

Randomness model

Deterministic

Input

Information vector $\mathbf{x} = (x_1, \ldots, x_k)$. Node $v_i \in \mathcal{V}$ possesses subset described by indices $\mathcal{Z}_{v_i}$.

Goal

Node $v_i \in \mathcal{V}$ requests subset of information vector described by indices $\mathcal{T}_{v_i}$.

Offline phase

The oracle finds $\mathbf{Y} \in \mathbf{P}$ which minimizes (4.5) such that (4.6) holds for every $v_i \in \mathcal{V}$. For every node $v_i \in \mathcal{V}$, the oracle defines encoding functions E_{v_i} as in (4.9) and for every requested index $j \in \mathcal{Z}_{v_i}$ the oracle defines decoding functions $D_{v_i,j}$ as in (4.8).

Online phase

$r = 1$

Computation phase

1. Every node v_i computes packet $\mathbf{p}_{v_i} = E_{v_i}(\mathcal{S}_{v_i})$.

Transmission phase

2. Every node v_i transmits $\mathbf{p}_{v_i}$ to out-neighbors in $\mathcal{W}_{\mathtt{out}(v_i)}$.

Recovery phase

3. Every node v_i uses received packets $(\mathbf{p}_{v_\ell})_{v_\ell \in \mathcal{W}_{\mathtt{in}(v_i)}}$ to compute the requested element for all $j \in \mathcal{T}_{v_i}$ as $x_j = D_{v_i,j}(\mathcal{S}_{v_i}, (\mathbf{p}_{v_\ell})_{v_\ell \in \mathcal{W}_{\mathtt{in}(v_i)}})$.

The corresponding information and query matrices are

$$\mathbf{Q}_{v_1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \mathbf{Q}_{v_2} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{Q}_{v_3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

$$\mathbf{Q}_{v_4} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \mathbf{Q}_{v_5} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{T}_{v_3} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{T}_{v_4} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{T}_{v_5} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

and $\mathbf{T}_{v_1} = \mathbf{T}_{v_2} = \mathbf{Z}$.

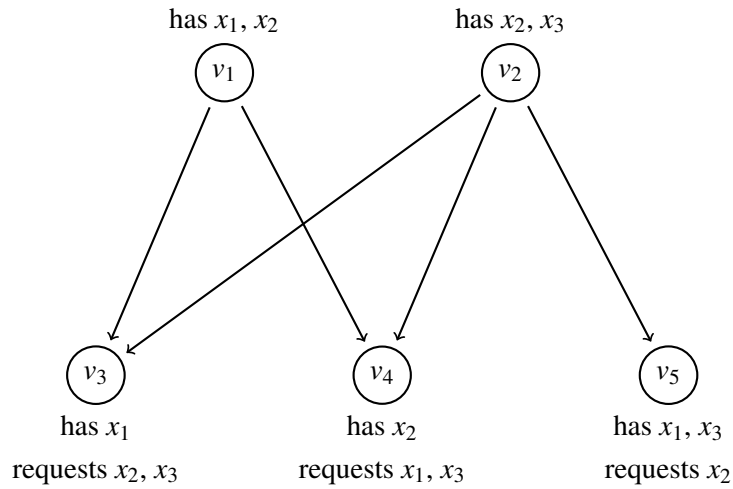

Figure 12. Example of an 1-solvable network

By choosing $\mathbf{Y} \in \mathbf{P}$ such that

$$\mathbf{Y}_{v_1} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \mathbf{Y}_{v_2} = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \mathbf{Y}_{v_3} = \mathbf{Y}_{v_4} = \mathbf{Y}_{v_5} = \mathbf{Z}.$$

For these choices, we can confirm that the following inclusions hold:

$$\text{rowspace} \begin{bmatrix} \mathbf{Y}_{v_1} \\ \mathbf{Y}_{v_2} \\ \mathbf{Q}_{v_3} \end{bmatrix} \supseteq \text{rowspace}(\mathbf{T}_{v_3}),$$

$$\text{rowspace} \begin{bmatrix} \mathbf{Y}_{v_1} \\ \mathbf{Y}_{v_2} \\ \mathbf{Q}_{v_4} \end{bmatrix} \supseteq \text{rowspace}(\mathbf{T}_{v_4}),$$

$$\text{rowspace} \begin{bmatrix} \mathbf{Y}_{v_2} \\ \mathbf{Q}_{v_5} \end{bmatrix} \supseteq \text{rowspace}(\mathbf{T}_{v_5}).$$

This means that the constraints (4.6) hold. We can verify through iterative search that such matrix $\mathbf{Y}$ has also minimal rank over all matrices in $\mathbf{P}$. By Theorem 18 we have that there exists a protocol for allowing all nodes to recover their requests and that it is optimal. The transmissions corresponding to $\mathbf{Y}$ are $x_1 + x_2$ by the node v_2 and $x_2 + x_3$ by the node v_2.

■

4.1.2. Arbitrary networks

In this section, we consider a more general scenario. We allow the underlying network topology to be any strongly connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The information

vector is $\mathbf{x} \in \mathbb{F}^k$ and every node $v \in \mathcal{V}$ possess a subset of it described by indices $\mathcal{Z}_v$ and requests all missing elements, i.e. $\mathcal{T}_v = [k] \setminus \mathcal{Z}_v$.

Even though the graph in Section 4.1.1 is allowed to be arbitrary, it is required that the data distribution problem instance has to be 1-solvable as in Definition 6. In this section, we consider instances, which are not 1-solvable.

Next, we give a very simple lower bound on the number of transmissions in Lemma 21.

Lemma 21. *For any node $v_i \in \mathcal{V}$ and for any $j \in [k]$, we denote by $\mu_{v_i}(j)$ the shortest path from any node $v_\ell \in \mathcal{V}$ to v_i such that $j \in \mathcal{Z}_{v_\ell}$:*

$$\mu_{v_i}(j) \triangleq \min_{\substack{\mathbf{t}_{v_i,v_\ell} \in \mathcal{L}_{v_i,v_\ell} \\ j \in \mathcal{Z}_{v_\ell}}} |\mathbf{t}_{v_i,v_\ell}|,$$

Furthermore, we denote

$$\mu_{v_i} \triangleq \sum_{j \in \mathcal{T}_{v_i}} \mu_{v_i}(j)$$

and

$$\mu_{\max} \triangleq \max_{v_i \in \mathcal{V}} \mu_{v_i}.$$

Then, the minimum number of transmissions in any protocol for data distribution which satisfies the requests of all nodes is at least $\mu_{\max}$.

Proof. Let v_ℓ be the node which maximizes $\mu_{\max}$. We see that the value μ_{v_ℓ} denotes the minimum number of transmissions which allows to recover the requests of v_ℓ. As there may be additional transmissions for enabling other nodes to recover their requests, then $\mu_{\max} = \mu_{v_\ell}$ yields a lower bound on the number of transmissions. □

We use the definition of sets of matrices $\mathbf{P}_{v_i} \subseteq \mathbf{S}^{\mathbb{F}}_{k\times k}$, $v_i \in \mathcal{V}$ as in (4.1) and $\mathbf{P} \subseteq \mathbf{S}^{\mathbb{F}}_{uk\times k}$ as in (4.2). However, we give an alternative representation which allows us to apply algebraic operations on the sets of matrices. For that, we define a symbol $\star$ which can take any value in the field $\mathbb{F}$. Now, for node $v_i \in \mathcal{V}$, we define the set of matrices $\mathbf{P}_{v_i}$ as a $k \times k$-dimensional matrix over $\mathbb{F}^\star \triangleq \mathbb{F} \cup \{\star\}$ such that

$$(\mathbf{P}_{v_i})_{j,\ell} = \begin{cases} \star & \text{if } \ell \in \mathcal{Z}_{v_i} \\ 0 & \text{otherwise} \end{cases} \tag{4.13}$$

and $\mathbf{P}$ as a concatenation of $\mathbf{P}_{v_i}$ in matrix form

$$\mathbf{P} \triangleq \begin{bmatrix} \mathbf{P}_{v_1} \\ \vdots \\ \mathbf{P}_{v_u} \end{bmatrix} \tag{4.14}$$

Now, we give some definitions for the matrix representation of the set of matrices.

Definition 8. The maximum rank of the set of matrices $\mathbf{P}$ is defined as

$$\text{max-rank}(\mathbf{P}) \triangleq \max_{\mathbf{Y}\in\mathbf{P}} \text{rank}(\mathbf{Y}).$$

Definition 9. Let $\mathbf{P}_{v_i}$ be a set of matrices as in (4.13). The operator $\Gamma()$ takes as input $\mathbf{P}_{v_i}$ and outputs a matrix $\mathbf{N}_{v_i} \in \mathbf{P}_{v_i}$ where

$$(\mathbf{N}_{v_i})_{j,\ell} = \begin{cases} 1 & \text{if } j = \ell \text{ and } (\mathbf{P}_{v_i})_{j,\ell} = \star \\ 0 & \text{otherwise} \end{cases}.$$

Furthermore, the operator $\Gamma_{v_i}()$ takes as input $\mathbf{P}$ as defined in (4.14) and outputs $\Gamma(\mathbf{P}_{v_i})$.

Essentially, Definition 9 generalizes the description of information matrix in (4.3).

Example 14. Let $\mathbf{P}_{v_i}$ be

$$\mathbf{P}_{v_i} = \begin{bmatrix} \star & 0 & \star & \star \\ \star & 0 & \star & \star \\ \star & 0 & \star & \star \\ \star & 0 & \star & \star \end{bmatrix}.$$

Then,

$$\Gamma(\mathbf{P}_{v_i}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

■

Definition 10. Let $x, y \in \mathbb{F}^\star$. Then the sum of x and y is defined as

$$x + y \triangleq \begin{cases} \star & \text{if } x = \star \text{ or } y = \star \\ x + y & \text{otherwise} \end{cases}.$$

Definition 11. Let $x, y \in \mathbb{F}^\star$. Then the product of x and y is defined as

$$x \cdot y \triangleq \begin{cases} 0 & \text{if } x = 0 \text{ or } y = 0 \\ \star & \text{if } (x = \star, y \neq 0) \text{ or } (x \neq 0, y = \star) \\ x \cdot y & \text{otherwise} \end{cases}.$$

The addition and multiplication of matrices over $\mathbb{F}^\star$ corresponds to usual matrix addition and multiplication where the element-wise operations are performed as in Definitions (10) and (11), while keeping in mind that the resulting matrix over $\mathbb{F}^\star$ represents a set of matrices.

In what follows, we use a binary operation of matrix multiplication, where one of the arguments is an integer matrix and the second argument is a family of matrices over $\mathbb{F}$, and the result is a family of matrices over $\mathbb{F}$. In order to be able

to do so, by slightly abusing the notation, we use the product of an integer matrix with a matrix over $\mathbb{F}^\star$, according to the rules defined in Definitions 10 and 11. The result of this operation is a matrix over $\mathbb{F}^\star$, which can be interpreted as a family of matrices over $\mathbb{F}$.

Example 15. Let a 3×3 matrix $\mathbf{M}$ over $\mathbb{F}$ and a 3×3 matrix family $\mathbf{P}$ over $\mathbb{F}$ be given by

$$\mathbf{M} = \begin{bmatrix} 1 & 2 & 0 \\ 4 & 5 & 6 \\ 0 & 7 & 8 \end{bmatrix} \text{ and } \mathbf{P} = \begin{bmatrix} \star & 0 & 0 \\ 0 & 0 & \star \\ 0 & \star & 0 \end{bmatrix}.$$

Multiplying $\mathbf{M}$ by $\mathbf{P}$ yields

$$\mathbf{MP} = \begin{bmatrix} 1 & 2 & 0 \\ 4 & 5 & 6 \\ 0 & 7 & 8 \end{bmatrix} \begin{bmatrix} \star & 0 & 0 \\ 0 & 0 & \star \\ 0 & \star & 0 \end{bmatrix} = \begin{bmatrix} \star & 0 & \star \\ \star & \star & \star \\ 0 & \star & \star \end{bmatrix}.$$

■

As $\mathbf{P}_{v_i}$ describes the elements in the possession of node v_i at a specific round of the protocol run, it is useful to give a relation for the possible elements in node's possession in the next round. The following Lemma describes the relationship between the elements in consecutive rounds.

Lemma 22. *Let* $\mathbf{P}$ *be the matrix over* $\mathbb{F}^\star$ *describing the elements possessed by nodes as in* (4.14)*. Let* $\mathbf{D}$ *be the adjacency matrix of the graph* $\mathcal{G}$ *as defined in Definition 7. Let* $\mathbf{E}$ *be an* $k \times k$ *all-one matrix. There exist encoding functions* E_{v_i} *and decoding functions* D_{v_i} *for every node* v_i *such that the matrix* $\mathbf{P}^+$ *describing the elements possessed by nodes after a round is related to* $\mathbf{P}$ *as*

$$\mathbf{P}^+ = (\mathbf{D} \otimes \mathbf{E}) \cdot \mathbf{P}. \tag{4.15}$$

Proof. The matrix $\mathbf{P}$ can be partitioned into $\mathbf{P}_{v_i}$, $v_i \in \mathcal{V}$ due to (4.14). From (4.13) we see that every $\mathbf{P}_{v_i}$ has k identical rows. Thus, we can write

$$\mathbf{P}_{v_i} = \mathbf{P}_{v_i}^{[1]} \otimes \mathbf{1}^\top,$$

where $\mathbf{1}^\top$ is a column vector of length k containing only ones.

By defining

$$\hat{\mathbf{P}} \triangleq \begin{bmatrix} \mathbf{P}_{v_1}^{[1]} \\ \vdots \\ \mathbf{P}_{v_u}^{[1]} \end{bmatrix},$$

we can thus write

$$\mathbf{P} = \hat{\mathbf{P}} \otimes \mathbf{1}^\top.$$

By using the properties of tensor product, we can write the right-hand side of (4.15) as

$$(\mathbf{D} \otimes \mathbf{E}) \cdot \mathbf{P} = (\mathbf{D} \cdot \hat{\mathbf{P}}) \otimes \mathbf{1}^\top. \tag{4.16}$$

By using the addition and multiplication rules as defined in Definitions 10 and 11, we have that

$$(\mathbf{D}\cdot\hat{\mathbf{P}})_{\ell,j} = \sum_{i\in[u]} (\mathbf{D})_{\ell,i}(\hat{\mathbf{P}})_{i,j},$$

where we have $(\mathbf{D}\cdot\hat{\mathbf{P}})_{\ell,j} = \star$ when there is an edge $(v_i, v_\ell) \in \mathcal{E}$ such that node v_i has $j \in \mathcal{Z}_{v_i}$. By defining E_{v_i} such that it includes x_j as a transmitted packet and D_{v_ℓ} such that it returns all received packets, then v_ℓ can recover the element x_j, i.e. we have that $j \in \mathcal{Z}^+_{v_\ell}$ after the protocol run.

Thus, we can conclude that the matrix $(\mathbf{D}\cdot\hat{\mathbf{P}})\otimes\mathbf{1}^\top$ correctly represents the elements every node has after the protocol run and due to (4.16) the claim (4.15) holds. □

By using induction principle, we can extend the result of Lemma 22 over any number of rounds of the protocol run. For that, we write the matrix $\mathbf{P}$ in round r as $\mathbf{P}^{(r)}$. We further use the notation $\mathbf{P}^{(0)} = \mathbf{P}$, where the matrix $\mathbf{P}$ describes the elements possessed by nodes initially.

Corollary 23. *Let $\mathbf{P}^{(0)}$ be the matrix over $\mathbb{F}^\star$ which describes the elements possessed by the nodes before the protocol runs. After r rounds of protocol run the matrix $\mathbf{P}^{(r)}$ describing the elements possessed by nodes is related to $\mathbf{P}^{(0)}$ as*

$$\mathbf{P}^{(r)} = (\mathbf{D}^r \otimes \mathbf{E})\cdot\mathbf{P}^{(0)}. \tag{4.17}$$

Proof. For matrix tensor product, the following property holds:

$$(\mathbf{M}\otimes\mathbf{N})\cdot(\mathbf{V}\otimes\mathbf{W}) = (\mathbf{M}\cdot\mathbf{V})\otimes(\mathbf{N}\cdot\mathbf{W}).$$

This allows us to apply Lemma 22 iteratively for $\mathbf{P}^{(r)}$:

$$\begin{aligned}\mathbf{P}^{(r)} &= (\mathbf{D}\otimes\mathbf{E})\cdot\mathbf{P}^{(r-1)}\\ &= (\mathbf{D}^r\otimes\mathbf{E}^r)\cdot\mathbf{P}^{(0)}.\end{aligned} \tag{4.18}$$

By observing that for a $k\times k$-dimensional ones matrix $\mathbf{E}$ we have

$$\mathbf{E}^r = k^{r-1}\mathbf{E},$$

we can write (4.18) as

$$\begin{aligned}(\mathbf{D}^r\otimes\mathbf{E}^r)\cdot\mathbf{P}^{(0)} &= (\mathbf{D}^r\otimes k^{r-1}\mathbf{E})\cdot\mathbf{P}^{(0)}\\ &= k^{r-1}(\mathbf{D}^r\otimes\mathbf{E})\cdot\mathbf{P}^{(0)}.\end{aligned}$$

Now, as $\mathbf{P}^{(0)}$ has only zero or symbol $\star$ in every cell, then also $(\mathbf{D}^r\otimes\mathbf{E})\cdot\mathbf{P}^{(0)}$ has also only zero or symbol $\star$ in every cell due to Definitions 10 and 11. Multiplying zero by k^{r-1} is zero and multiplying symbol $\star$ is $\star$. Thus we can omit k^{r-1} and get

$$k^{r-1}(\mathbf{D}^r\otimes\mathbf{E})\cdot\mathbf{P}^{(0)} = (\mathbf{D}^r\otimes\mathbf{E})\cdot\mathbf{P}^{(0)}$$

as claimed. □

We can now state the main result of this section. The following Theorem 24 generalizes the result of Theorem 18 to any network topology where there exists a protocol for data distribution.

Theorem 24. *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the graph describing the network topology of a data distribution problem, where every node $v_i \in \mathcal{V}$ possesses subset of the information vector $\mathbf{x} = (x_1, \ldots, x_k)$ described by the indices $\mathcal{Z}_{v_i} \subseteq [k]$ and requests missing elements described by indices $\mathcal{T}_{v_i} = [k] \setminus \mathcal{Z}_{v_i}$.*

Let the graph be ρ_0-solvable for some $\rho_0 \in \mathbb{N}$. Let the adjacency matrix of the graph $\mathcal{G}$ be $\mathbf{D}$ and let the set of matrices $\mathbf{P}$ describe the elements possessed by the nodes at round 0.

Then, for any $\rho \geq \rho_0$, there exist encoding and decoding functions $E_{v_i}^{(r)}$ and $D_{v_i}^{(r)}$, $v_i \in \mathcal{V}$, $r \in [\rho]$, such that at the end of the protocol every node has recovered requested elements. Furthermore, the total number of transmitted packets for such defined encoding and decoding functions is

$$\pi = \sum_{r=1}^{\rho} \min_{\mathbf{Y}^{(r)} \in (\mathbf{D}^{(r-1)} \otimes \mathbf{E}) \cdot \mathbf{P}} \sum_{v_i \in \mathcal{V}}^{u} \operatorname{rank}(\mathbf{Y}_{v_i}^{(r)}) \tag{4.19}$$

for matrices $\mathbf{Y}^{(r)}$ which for every $v_i \in \mathcal{V}$ are subject to constraints

$$\operatorname{rank}\left(\begin{bmatrix} (\operatorname{diag}(\mathbf{D}^{[v_i]}) \otimes \mathbf{I}) \cdot \mathbf{Y}^{(r)} \\ \Gamma_{v_i}((\mathbf{D}^{r-1} \otimes \mathbf{E}) \cdot \mathbf{P}) \end{bmatrix}\right) = \text{max-rank}((\operatorname{diag}(\mathbf{e}_{v_i}) \otimes \mathbf{I}) \cdot (\mathbf{D}^{(r)} \otimes \mathbf{E}) \cdot \mathbf{P}), \tag{4.20}$$

where matrices $\mathbf{I}$ and $\mathbf{E}$ are $u \times u$-dimensional.

We emphasize that Theorem 24 is an existence result, and it does not necessarily imply that the number of packets in (4.19) is optimal.

Before we prove Theorem 24, we state and prove a technical result used for the proof.

Lemma 25. *Let $\mathcal{G}$ be a graph defined by the adjacency matrix $\mathbf{D}$. Let the set of matrices $\mathbf{P}$ describe the elements possessed by nodes as in* (4.14)*. Then, there exists a matrix $\mathbf{Y} \in \mathbf{P}$ such that for every $v_i \in \mathcal{V}$*

$$\operatorname{rank}\left(\begin{bmatrix} (\operatorname{diag}(\mathbf{D}^{[v_i]}) \otimes \mathbf{I}) \cdot \mathbf{Y} \\ \Gamma_{v_i}(\mathbf{P}) \end{bmatrix}\right) = \text{max-rank}((\operatorname{diag}(\mathbf{e}_{v_i}) \otimes \mathbf{I}) \cdot (\mathbf{D} \otimes \mathbf{E}) \cdot \mathbf{P}). \tag{4.21}$$

Proof. We consider the left-hand and right-hand side of (4.21) separately for every $v_i \in \mathcal{V}$.

Right-hand side: We consider the matrix $(\operatorname{diag}(\mathbf{e}_{v_i}) \otimes \mathbf{I}) \cdot (\mathbf{D} \otimes \mathbf{E}) \cdot \mathbf{P}$ in the max-rank of (4.21). First, similarly as in the proof of Lemma 22, we can write

$$\mathbf{P} = \hat{\mathbf{P}} \otimes \mathbf{1}^{\top},$$

and thus

$$(\operatorname{diag}(\mathbf{e}_{v_i}) \otimes \mathbf{I}) \cdot (\mathbf{D} \otimes \mathbf{E}) \cdot \mathbf{P} = (\operatorname{diag}(\mathbf{e}_{v_i}) \otimes \mathbf{I}) \cdot (\mathbf{D} \otimes \mathbf{E}) \cdot (\hat{\mathbf{P}} \otimes \mathbf{1}^{\top})$$

$$
\begin{aligned}
&= (\text{diag}(\mathbf{e}_{v_i}) \cdot \mathbf{D} \cdot \hat{\mathbf{P}}) \otimes (\mathbf{I} \cdot \mathbf{E} \cdot \mathbf{1}^\top) \\
&= (\text{diag}(\mathbf{e}_{v_i}) \cdot \mathbf{D} \cdot \hat{\mathbf{P}}) \otimes (u\mathbf{1}^\top) \\
&= u(\text{diag}(\mathbf{e}_{v_i}) \cdot \mathbf{D} \cdot \hat{\mathbf{P}}) \otimes \mathbf{1}^\top. \qquad (4.22)
\end{aligned}
$$

As $u > 0$, we can omit the scalar. (4.22) becomes

$$
(\text{diag}(\mathbf{e}_{v_i}) \otimes \mathbf{I}) \cdot (\mathbf{D} \otimes \mathbf{E}) \cdot \mathbf{P} = (\text{diag}(\mathbf{e}_{v_i}) \cdot \mathbf{D} \cdot \hat{\mathbf{P}}) \otimes \mathbf{1}^\top.
$$

The only non-zero row of $u \times k$-dimensional matrix $\text{diag}(\mathbf{e}_{v_i}) \cdot \mathbf{D} \cdot \hat{\mathbf{P}}$ is the v_i-th row. The values in the row for all $\ell \in [k]$ are

$$
(\text{diag}(\mathbf{e}_{v_i}) \cdot \mathbf{D} \cdot \hat{\mathbf{P}})_{v_i,\ell} = \sum_{v_j \in \mathcal{V}} (\mathbf{D})_{v_i,v_j} (\hat{\mathbf{P}})_{v_j,\ell}. \qquad (4.23)
$$

As $\left|\mathbf{1}^\top\right| = k$, then max-rank$((\text{diag}(\mathbf{e}_{v_i}) \cdot \mathbf{D} \cdot \hat{\mathbf{P}}) \otimes \mathbf{1}^\top)$ is the number of symbols $\star$ in the only non-zero row of $\text{diag}(\mathbf{e}_{v_i}) \cdot \mathbf{D} \cdot \hat{\mathbf{P}}$.

Left-hand side: First, we look at $(\text{diag}(\mathbf{D}^{[v_i]}) \otimes \mathbf{I}) \cdot \mathbf{Y}$ and show how to choose the values in $\mathbf{Y}$ such that the equality in (4.21) holds.

First, the matrix $\text{diag}(\mathbf{D}^{[v_i]}) \otimes \mathbf{I}$ is a block-diagonal matrix

$$
\begin{bmatrix}
\mathbf{D}_{v_i,1} & \mathbf{Z} & \cdots & \mathbf{Z} \\
\mathbf{Z} & \mathbf{D}_{v_i,2} & \cdots & \mathbf{Z} \\
\vdots & \vdots & \ddots & \vdots \\
\mathbf{Z} & \mathbf{Z} & \cdots & \mathbf{D}_{v_i,u},
\end{bmatrix},
$$

where $\mathbf{D}_{v_i,\ell}$, $\ell \in [u]$, is a $k \times k$-dimensional matrix with value $(\mathbf{D})_{v_i,\ell}$ in every cell in the main diagonal.

Then, we have that

$$
(\text{diag}(\mathbf{D}^{[v_i]}) \otimes \mathbf{I}) \cdot \mathbf{Y} = \begin{bmatrix}
(\mathbf{D})_{v_i,1}(\mathbf{Y})_{1,1} & \cdots & (\mathbf{D})_{v_i,1}(\mathbf{Y})_{1,k} \\
\vdots & \cdots & \vdots \\
(\mathbf{D})_{v_i,1}(\mathbf{Y})_{k,1} & \cdots & (\mathbf{D})_{v_i,1}(\mathbf{Y})_{k,k} \\
\vdots & \cdots & \vdots \\
(\mathbf{D})_{v_i,u}(\mathbf{Y})_{(u-1)k+1,1} & \cdots & (\mathbf{D})_{v_i,u}(\mathbf{Y})_{(u-1)k+1,k} \\
\vdots & \cdots & \vdots \\
(\mathbf{D})_{v_i,u}(\mathbf{Y})_{uk,1} & \cdots & (\mathbf{D})_{v_i,u}(\mathbf{Y})_{uk,k}
\end{bmatrix}. \qquad (4.24)
$$

We now describe, how to set the values in the matrix $\mathbf{Y}$ such that $\mathbf{Y} \in \mathbf{P}$ and the equality in (4.21) holds. We see from (4.23) that in the matrix $\text{diag}(\mathbf{e}_{v_i}) \cdot \mathbf{D} \cdot \hat{\mathbf{P}}$ the value in the v_i-th row and ℓ-th column is $\star$ if there exists a node $v_j \in \mathcal{V}$ such that there is an edge $(v_j, v_i) \in \mathcal{E}$ and $\ell \in \mathcal{Z}_{v_j}$.

As $\left|\mathcal{Z}_{v_j}\right| \leq k$, then we can choose different $s \in [k]$ for every $\ell \in \mathcal{Z}_{v_j}$ and set the cell $(\mathbf{Y})_{(v_j-1)k+s,\ell}$ to 1. After setting ones in $\mathbf{Y}$, we set all other cells to 0 in $\mathbf{Y}$.

Thus, as the ones in matrix (4.24) are all in distinct row and columns, the rank of it equals the number of ones. As the number of ones in (4.24) is the number of symbols $\star$ in (4.23), we have

$$\text{rank}((\text{diag}(\mathbf{D}^{[v_i]}) \otimes \mathbf{I}) \cdot \mathbf{Y}) = \text{max-rank}((\text{diag}(\mathbf{e}_{v_i}) \otimes \mathbf{I}) \cdot (\mathbf{D} \otimes \mathbf{E}) \cdot \mathbf{P}). \quad (4.25)$$

We also see that as we set only such cells to 1 in $\mathbf{Y}$ where in the corresponding cell in $\mathbf{P}$ has value $\star$, then $\mathbf{Y} \in \mathbf{P}$.

Now, we show that when we extend the matrix $(\text{diag}(\mathbf{D}^{[v_i]}) \otimes \mathbf{I}) \cdot \mathbf{Y}$ by $\Gamma_{v_i}(\mathbf{P})$, the rank does not change.

Due to the construction of $\mathbf{Y} \in \mathbf{P}$, the matrix $(\text{diag}(\mathbf{D}^{[v_i]}) \otimes \mathbf{I}) \cdot \mathbf{Y}$ has no more than a single one in every row. The ones are in columns where there is $\star$ in the same columns of $(\text{diag}(\mathbf{e}_{v_i}) \otimes \mathbf{I}) \cdot (\mathbf{D} \otimes \mathbf{E}) \cdot \mathbf{P}$. As the adjacency matrix $\mathbf{D}$ has ones in its main diagonal, then if there is a column with $\star$ in $(\text{diag}(\mathbf{e}_{v_i}) \otimes \mathbf{I}) \cdot (\mathbf{D} \otimes \mathbf{E}) \cdot \mathbf{P}$, there must also exist a row in $\mathbf{P}$ with $\star$ in the same column.

Due to Definition 9, this means that if there is a single one in a row in any column of $(\text{diag}(\mathbf{D}^{[v_i]}) \otimes \mathbf{I}) \cdot \mathbf{Y}$, then there is also a single one in a row in the corresponding column of Γ_{v_i}. Due to (4.25), we have

$$\text{rowspace}(\Gamma_{v_i}(\mathbf{P})) \subseteq \text{rowspace}((\text{diag}(\mathbf{D}^{[v_i]}) \otimes \mathbf{I}) \cdot \mathbf{Y})$$

and thus

$$\text{rank}\left(\begin{bmatrix}(\text{diag}(\mathbf{D}^{[v_i]}) \otimes \mathbf{I}) \cdot \mathbf{Y} \\ \Gamma_{v_i}(\mathbf{P})\end{bmatrix}\right) = \text{rank}((\text{diag}(\mathbf{D}^{[v_i]}) \otimes \mathbf{I}) \cdot \mathbf{Y}),$$

completing the proof of the claim (4.21). □

We remark that Lemma 25 claims that there is such matrix $\mathbf{Y} \in \mathbf{P}$, but does not imply that the constructed matrix is optimal.

Proof of Theorem 24. From Lemma 25, there exist matrices $\mathbf{Y}^{(r)}$ for every round $r \in [\rho]$ which satisfies the condition (4.20). As in (4.14), we can partition $\mathbf{Y}^{(r)}$ as $\mathbf{Y}^{(r)}_{v_i}$, $v_i \in \mathcal{V}$. Without loss of generality, we can assume that only the first

$$\pi^{(r)}_{v_i} \triangleq \text{rank}(\mathbf{Y}^{(r)}_{v_i})$$

rows of $\mathbf{Y}^{(r)}_{v_i}$ are non-zero. We take the first $\pi^{(r)}_{v_i}$ rows as linear coefficients for encoding the transmitted packets, i.e. the encoding function is

$$E^{(r)}_{v_i}(\mathcal{Z}^{(r-1)}_{v_i}) \triangleq \left((\mathbf{Y}^{(r)}_{v_i})^{[j]} \cdot \mathbf{x}\right)_{j \in [\pi^{(r)}_{v_i}]}.$$

We indeed see that the encoding function is correctly defined as $\mathbf{Y}^{(r)}_{v_i} \in \mathbf{P}^{(r-1)}_{v_i}$, for $\mathbf{P}^{(r-1)}_{v_i}$ as defined in Corollary 23 and thus uses values available to node v_i in that round.

Additionally, as the number of the transmitted elements is $\pi_{v_i}^{(r)}$, then by summing up over all nodes, we obtain that the number of the transmitted packets in the round r is

$$\pi^{(r)} \triangleq \sum_{v_i \in \mathcal{V}} \pi_{v_i}^{(r)}. \tag{4.26}$$

We still have to ensure that every node v_i can recover all the elements. However, as the encoding function corresponds to the encoding functions defined in Lemma 22, we have that there is a decoding function such that the elements possessed by the nodes after the round r is described by the matrix $(\mathbf{D}^r \otimes \mathbf{E}) \cdot \mathbf{P}$.

Finally, by considering the matrices which minimize the number of transmissions (4.26) in every round and summing up over all rounds, we get that the total number of packets transmitted in the protocol is as given in (4.19). □

Similarly to the remark in Section 4.1.1, the computation complexity of the protocol given by Theorem 24 is dominated by finding the matrices $\mathbf{Y}^{(r)}$, $r \in [\rho]$. In every round r, every node $v \in \mathcal{V}$ has to compute $\pi_v^{(r)}$ inner product to encode the packets and solve a system of linear equations, which can be done in cubic time in the number of variables.

The results in this Chapter are close to the results by Courtade and Wesel in [12] and Gonen and Langberg in [27]. Compared to [12], we do not require that the underlying network topology of 1-solvable network to be complete. For arbitrary networks, they require that the underlying network topology is regular, meaning that every node has the same number of neighbors. Furthermore, there is a restriction that the elements have to be divisible into partitions which is unnecessary for our results.

The paper [27] was published approximately at the same time as our similar results in [35]. Our current setting is close to the model in Section V-B in [27]. Their result is given as a reduction to Directed Weighted Steiner Tree, which is known to be NP-hard. As such, the results are comparable but based on reductions to different problems computationally difficult problems.

In order to conclude this section, we provide a full protocol description for data distribution problem in a network topology, where the underlying graph is a strongly connected graph. The protocol is given as Protocol 7.

4.2. Experimental results

We remark that the result of Theorem 24 may not be optimal in the number of transmitted packets. This is due to the greedy nature of the protocol as the packets are constructed in such a way that the nodes could recover missing elements in the minimal number of rounds.

In order to test the efficiency of Theorem 24, we ran numerical simulations to compare the number of packets obtained from the theorem to the lower bound in Lemma 21. For every run we generate a random graph described by its adjacency

Protocol 7 Data distribution in arbitrary strongly connected network

Label

Π_7

Network topology

Strongly connected graph with u nodes $\mathcal{V}$.

Randomness model

Deterministic

Input

Information vector $\mathbf{x} = (x_1, \ldots, x_k)$. Initially, node $v_i \in \mathcal{V}$ possesses a subset of the information vector described by the indices $\mathcal{Z}_{v_i}^{(0)} \subseteq [k]$

Goal

Node $v_i \in \mathcal{V}$ requests remaining elements $\mathcal{T}_{v_i} = [k] \setminus \mathcal{Z}_{v_i}^{(0)}$.

Offline phase

Oracle finds $\mathbf{Y}^{(r)} \in \mathbf{P}^{(r)}$ for every $r \in [\rho]$ for $\mathbf{P}^{(r)}$ defined in (4.17) such that (4.20) holds and number of transmitted packets is minimized as in (4.19). For every $\mathbf{Y}^{(r)}$, the oracle defines encoding and decoding functions $E_{v_i}^{(r)}$ and $D_{v_i}^{(r)}$ and transmits them to corresponding nodes $v_i \in \mathcal{V}$.

Online phase

For $r \in [\rho]$

Computation phase

1. Every node v_i computes packet $\mathbf{p}_{v_i}^{(r)} = E_{v_i}^{(r)}(\mathcal{Z}_{v_i}^{(r-1)})$.

Transmission phase

2. Every node v_i transmits $\mathbf{p}_{v_i}^{(r)}$ to out-neighbors in $\mathcal{W}_{\mathtt{out}(v_i)}$.

Recovery phase

3. Every node v_i uses received packets $(\mathbf{p}_{v_\ell}^{(r)})_{v_\ell \in \mathcal{W}_{\mathtt{in}(v_i)}}$ to recover the elements described by $\mathbf{P}_{v_i}^{(r)}$ as $\mathcal{K}_{v_i}^{(r)} = D_{v_i}^{(r)}(\mathcal{Z}_{v_i}^{(r)}, (\mathbf{p}_{v_\ell}^{(r)})_{v_\ell \in \mathcal{W}_{\mathtt{in}(v_i)}})$.
4. Every node v_i updates their sets $\mathcal{S}_{v_i}^{(r)} = \mathcal{K}_{v_i}^{(r)} \cup \mathcal{S}_{v_i}^{(r-1)}$ and corresponding indices set $\mathcal{Z}_{v_i}^{(r)}$.

matrix and randomly assign elements to the nodes. The graphs are generated in such a way that the longest path between any two nodes (i.e. the diameter of the graph) is fixed.

The number of nodes and information bits in every run is fixed at 4. The diameter of graph is either 2 or 3. We ran 1000 experiments for every choice of the parameters. Within every run, we computed the lower bound on the number of packets given by Lemma 21 and the number of packets given by Theorem 24 and computed the ratio.

As the computational complexity of finding the matrix which minimizes (4.19) is exponential in $k \sum_{v_i \in \mathcal{V}} |\mathcal{Z}_{v_i}|$, we instead perform an iterative search over the ma-

trices described by **P**. Additionally, for computation of the max-rank of a set of matrices, we use the algorithm described in [25].

The numerical simulation results are given in Table 11.

Table 11. The efficiency of Theorem 24 compared to Lemma 21

Ratio	$[1, 1.2)$	$[1.2, 1.4)$	$[1.4, 1.6)$	$[1.6, 1.8)$	$[1.8, 2.0)$	$[2, \infty)$
Cases (diam. 2)	54%	22%	6%	4%	0%	14%
Cases (diam. 3)	30%	18%	24%	0%	6%	22%

5. FUNCTION COMPUTATION ON SYNCHRONIZED DATA

In this chapter, we present protocols for computing particular functions over the union of the sets of vectors in the possession of the nodes. We show in Section 5.1.1 that the problem of computing such functions generalizes the data synchronization problem. We also show that the naive approach of first synchronizing the sets and then computing the value of function is sub-optimal.

In Section 5.2, we use F-monochromatic rectangles to bound from below the communication complexity of the protocols. The main results of the section are Theorems 28 and 29.

In Section 5.3, we use reduction to known protocols in order to compute the values of the given functions. Specifically, in Section 5.3.1 we use a reduction to set disjointness problem and in Section 5.3.2 we use a reduction from a set intersection problem. The main results are given as Lemmas 30 and 32.

In Section 5.4, we describe an error-free function for computing a particular function F and analyze its communication complexity. The main result is given as Theorem 33.

5.1. Problem definition

Let the underlying field in this chapter be $\mathbb{F} = \mathbb{F}_2$. We denote the vectors of length n over $\mathbb{F}$ as $\mathbb{F}^n$. Furthermore, the vectors in $\mathbb{F}^n$ allow to represent integers of bit length n, i.e. the range $[0, 2^n - 1]$.

In this chapter, we look at the networks which have only two nodes A and B. Both of the nodes possess a subset of the elements $\mathcal{S}_A, \mathcal{S}_B \subseteq \mathbb{F}^n$. We denote the intersection of the sets as $\mathcal{S}_0 \triangleq \mathcal{S}_A \cap \mathcal{S}_B$. We denote the sizes of the sets as

$$\begin{aligned} k_0 &\triangleq |\mathcal{S}_0|, \\ k_A &\triangleq |\mathcal{S}_A|, \\ k_B &\triangleq |\mathcal{S}_B|, \end{aligned}$$

and the number of unique elements as

$$\begin{aligned} d_A &\triangleq |\mathcal{S}_A \setminus \mathcal{S}_0|, \\ d_B &\triangleq |\mathcal{S}_B \setminus \mathcal{S}_0|, \\ d &\triangleq d_A + d_B. \end{aligned}$$

We assume that the node A knows values k_A and d_A and the node B knows k_B and d_B.

Additionally, we assume that the sizes of the sets are bounded by k:

$$\max\{k_A, k_B\} \leq k.$$

Instead of considering general functions, we instead restrict the functions to be in the form

$$F(\mathcal{S}_A, \mathcal{S}_B) = \Phi(\mathcal{S}_A \cup \mathcal{S}_B),$$

where $\mathcal{S}_A \cup \mathcal{S}_B$ is the standard set-union of the sets $\mathcal{S}_A$ and $\mathcal{S}_B$ and

$$\Phi : 2^{\mathbb{F}^n} \to \mathcal{Y}$$

is an arbitrary function. In essence, the nodes want to compute some function Φ on the union of their sets.

5.1.1. Connection to data synchronization

We can consider the data synchronization problem as a special case of the function computation problem on synchronized data, where the function Φ to be computed is an identity function

$$\Phi(\mathcal{S}) \triangleq \mathcal{S},$$

and thus we have that the nodes have to cooperatively compute the function F as

$$F(\mathcal{S}_A, \mathcal{S}_B) = \mathcal{S}_A \cup \mathcal{S}_B.$$

In Chapter 3 we saw that there exists optimal data synchronization protocols which achieve the communication complexity of $\mathcal{O}(dn)$. By using reduction to the data synchronization protocol, it is possible to construct a protocol for computing any function F by first letting the nodes to synchronize the elements with communication complexity $\mathcal{O}(dn)$, and then having the nodes to compute the value of corresponding function Φ on the synchronized data. Sometimes it may be beneficial if the synchronization happens only one-way, i.e. only a single node obtains the union of the sets, and sends the computed value to the other node.

However, we can construct a very trivial counter-example which shows that the approach of reducing function computation on synchronized data problem to data synchronization problem is not optimal in general. The counter-example is given as Example 16.

Example 16. Assume that the nodes A and B are interested in computing

$$F(\mathcal{S}_A, \mathcal{S}_B) = \max(\mathcal{S}_A \cup \mathcal{S}_B),$$

where all entries in $\mathcal{S}_A \cup \mathcal{S}_B$ are viewed as non-negative integer numbers in their binary representation. The following Protocol 8 requires only $2n$ bits of communication.

Protocol 8 Computing the maximum element over two sets

Label
Π_8

Network topology
Nodes $\mathcal{V} = \{A, B\}$, edges $\mathcal{E} = \{(A,B),(B,A)\}$.

Randomness model
Deterministic

Input
Nodes A and B have sets $\mathcal{S}_A$ and $\mathcal{S}_B$.

Goal
Nodes obtain $\max(\mathcal{S}_A \cup \mathcal{S}_B)$.

Online phase

$r = 1$

Computation phase

1. Node A computes $w_A = \max(\mathcal{S}_A)$.
2. Node B computes $w_B = \max(\mathcal{S}_B)$.

Transmission phase

3. Node A transmits w_A to node B.
4. Node B transmits w_B to node A.

Recovery phase

5. Nodes A and B recover $w = \max(\{w_A, w_B\})$.

We see that the total number of bits in Protocol 8 is $2n$ bits, as only a single

element from $\mathbb{F}^n$ is sent in step 3 and a single element in step 4. This dominates over all protocols which reduce to data synchronization. ■

It is possible to build analogous protocols for a number of other idempotent functions Φ, such as *minimum*, bit-wise logical *or* and bit-wise logical *and*.

5.2. Lower bounds on function computation on synchronized data using F-monochromatic rectangles

In this section, we consider a few arithmetic functions and establish lower bounds on the communication complexity in different randomness models using F-monochromatic rectangles as defined in [37].

Definition 12 ([37, Definition 1])**.** Let $\omega \in \mathbb{N}$ and $F : \mathbb{F}^\omega \times \mathbb{F}^\omega \to \mathcal{Y}$ be a function with range $\mathcal{Y}$. A rectangle is a subset of $\mathbb{F}^\omega \times \mathbb{F}^\omega$ of the form $\mathcal{S}_1 \times \mathcal{S}_2$, where $\mathcal{S}_1, \mathcal{S}_2 \subseteq \mathbb{F}^\omega$. A rectangle $\mathcal{S}_1 \times \mathcal{S}_2$ is called F-monochromatic if for every $x_1 \in \mathcal{S}_1$ and $x_2 \in \mathcal{S}_2$, the value of $F(x_1, x_2)$ is constant.

The following lemma indicates that rectangle from Definition 12 is similar to a geometric rectangle. Namely, if corners on one diagonal belong to the rectangle, then also the corners of another diagonal belong to the same rectangle.

Lemma 26 ([38, Proposition 1.13])**.** *Let $R \subseteq \mathbb{F}^\omega \times \mathbb{F}^\omega$. Then R is a rectangle if and only if*

$$(x_1, y_1) \in R \text{ and } (x_2, y_2) \in R \Rightarrow (x_1, y_2) \in R.$$

It was shown in [38], that the communication complexity of deterministic protocols can be given through the number of F-monochromatic rectangles. In essence, this allows to reformulate finding the lower bound on the communication complexity as a problem in combinatorics. In order to present Lemma 27, we first need the following Definition 13.

Definition 13. Let $F : \mathbb{F}^\omega \times \mathbb{F}^\omega \to \mathcal{Y}$ be a function. Denote by $\Upsilon(F)$ the minimum number of F-monochromatic rectangles that partition the whole space of $\mathbb{F}^\omega \times \mathbb{F}^\omega$.

Lemma 27 ([38, Corollary 1.17])**.** *Let $F : \mathbb{F}^\omega \times \mathbb{F}^\omega \to \mathcal{Y}$ be a function, which is computed using protocol Π. Then*

$$\mathrm{COMM}(\Pi) \geq \log_2(\Upsilon(F)). \tag{5.1}$$

The proof idea of Lemma 27 is that it is possible to construct a binary search tree for all F-monochromatic rectangles and the communicated bits in the protocol indicate the choices at every node to reach the destination leaf. As the height of the binary search tree is $\log_2(\Upsilon(F))$, we get the inequality (5.1).

In order to use Lemma 27, we need to represent the sets $\mathcal{S}_A$ and $\mathcal{S}_B$ as binary vectors. The natural way is to do this by using characteristic vectors of the sets. A characteristic vector of a set $\mathcal{S} \in \mathcal{X}$ is a binary vector of length $|\mathcal{X}|$ where the i-th bit is 1 if $x_i \in \mathcal{S}$. As the sets are from $\mathbb{F}^n$, then the length of canonical vectors is thus $\omega = 2^n$.

5.2.1. Sum over integers

In this section, we consider the function

$$F(\mathcal{S}_A, \mathcal{S}_B) \triangleq \sum_{x \in \mathcal{S}_A \cup \mathcal{S}_B} x. \tag{5.2}$$

The lower bound on any deterministic protocol computing the value of the function is given by the following Theorem 28.

Theorem 28. *The number of bits communicated between the nodes A and B in any deterministic protocol* Π *that computes the function F in* (5.2) *is at least*

$$\mathrm{COMM}(\Pi) \geq 2^n + n - 1.$$

Proof. We obtain the bound by counting the number of F-monochromatic rectangles.

We denote by

$$\mathcal{J} \triangleq \mathbb{F}^n \setminus \{0\}, \tag{5.3}$$

where the elements of $\mathcal{J}$ can be viewed as integers in $[2^n - 1]$. We use the following set of pairs of subsets

$$\begin{aligned} \mathcal{B}_0 &\triangleq \{(\mathcal{S}, \mathcal{J} \setminus \mathcal{S}) : \mathcal{S} \subseteq \mathcal{J}\} \\ &\triangleq \{(\mathcal{S}_i, \mathcal{S}_i') : i \in [2^{2^n-1}]\}, \end{aligned} \tag{5.4}$$

i.e. partitions of the domain $\mathbb{F}^n$ without the element 0. We omit 0 as this does not change the value of the summation.

Then, for every $(\mathcal{S}_i, \mathcal{S}_i') \in \mathcal{B}_0$ we have

$$F(\mathcal{S}_i, \mathcal{S}_i') = \sum_{j=1}^{2^n-1} j = 2^{n-1}(2^n - 1).$$

On the other hand, if we take $i, j \in [2^{2^n-2}]$, such that $i \neq j$, then we have two cases:

- If $\mathcal{S}_i \cup \mathcal{S}_j' \neq \mathcal{J}$, then there exists $x \in \mathcal{J}$, such that $x \notin \mathcal{S}_i \cup \mathcal{S}_j'$. In that case, clearly
$$F(\mathcal{S}_i, \mathcal{S}_j') < 2^{n-1}(2^n - 1).$$
- If $\mathcal{S}_i \cup \mathcal{S}_j' = \mathcal{J}$, since $\mathcal{S}_i \neq \mathcal{S}_j$, there exists $x \in \mathcal{S}_i \cap \mathcal{S}_j'$. Thus, $x \notin \mathcal{S}_i' \cup \mathcal{S}_j$, and therefore
$$F(\mathcal{S}_j, \mathcal{S}_i') < 2^{n-1}(2^n - 1).$$

Therefore, due to Lemma 26, there are at least 2^{2^n-1} different F-monochromatic rectangles consisting of elements of $\mathcal{B}_0$.

Additionally, for any $\ell \in [2^n - 1]$, denote

$$\mathcal{J}_\ell \triangleq \mathbb{F}^n \setminus \{0, \ell\}.$$

We use the following pairs

$$\begin{aligned}\mathcal{B}_\ell &\triangleq \{(\mathcal{S}, \mathcal{J}_\ell \setminus \mathcal{S}) : \mathcal{S} \subseteq \mathcal{J}_\ell\} \\ &= \{(\mathcal{S}_i, \mathcal{S}'_i) : i \in [2^{2^n-2}]\},\end{aligned} \tag{5.5}$$

i.e. partitions of the domain $\mathbb{F}^n$ without elements 0 and ℓ.

Then, for every $(\mathcal{S}_i, \mathcal{S}'_i) \in \mathcal{B}_\ell$ we have

$$F(\mathcal{S}_i, \mathcal{S}'_i) = \sum_{j=1}^{2^n-1} j - \ell = 2^{n-1}(2^n-1) - \ell.$$

On the other hand, by taking $i, j \in [2^{2^n-1}]$, such that $i \neq j$, then similarly for the pairs $\mathcal{B}_0$ it can be shown that either

$$F(\mathcal{S}_i, \mathcal{S}'_j) < 2^{n-1}(2^n-1) - \ell$$

or

$$F(\mathcal{S}_j, \mathcal{S}'_i) < 2^{n-1}(2^n-1) - \ell.$$

Therefore, due to Lemma 26, there are at least 2^{2^n-2} different F-monochromatic rectangles consisting of the elements of $\mathcal{B}_\ell$. Since ℓ can be chosen 2^n-1 ways, we conclude that the number of different F-monochromatic rectangles is at least

$$\begin{aligned}\Upsilon(F) &\geq |\mathcal{B}_0| + (2^n-1)\,|\mathcal{B}_\ell| \\ &= 2^{2^n-1} + (2^n-1)\cdot(2^{2^n-2}) \\ &= (2^{2^n-2})\cdot(2^n+1) \\ &> 2^{2^n+n-2}.\end{aligned}$$

Finally, by applying Lemma 27, and by rounding the result up to the next bit, we obtain that the communication complexity of any protocol Π computing function (5.2) is at least

$$\mathrm{COMM}(\Pi) \geq 2^n + n - 1.$$

□

The following Example 17 illustrates the F-monochromatic rectangles in the proof of Theorem 28.

Example 17. In Figure 13, we show the different F-monochromatic rectangles whose existence is proved in Theorem 28 for $n = 2$. In this case, we have that $\mathcal{X} = \{0,1,2,3\}$. On the axes, we have the characteristic vectors of all sets which exclude element 0 (due to definition of $\mathcal{J}$ in (5.3)). The pairs $\mathcal{B}_0$ as defined in (5.4) are filled with gray, pairs $\mathcal{B}_1$ as defined in (5.5) have solid edge, pairs $\mathcal{B}_2$ are inverted and pairs $\mathcal{B}_3$ have dotted edge.

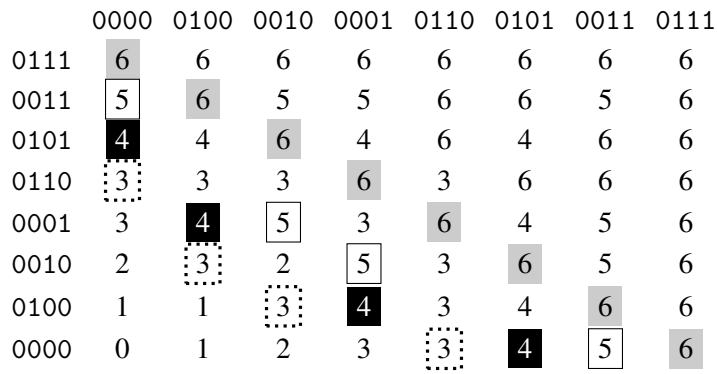

Figure 13. Example of f-monochromatic rectangles in the proof of Theorem 28 for $n = 2$

As described in the proof, every painted cell in the Figure represents a single F-monochromatic rectangle. We see that the total number of F-monochromatic rectangles is at least

$$\begin{aligned}\Upsilon(F) &\geq |\mathcal{B}_0| + |\mathcal{B}_1| + |\mathcal{B}_2| + |\mathcal{B}_3| \\ &= 8 + 4 + 4 + 4 \\ &= 20\end{aligned}$$

Due to Lemma 27, the communication complexity is at least

$$\log_2(\Upsilon(F)) = \log_2(20)$$

bits. By rounding up to the next integer, we obtain that

$$\mathrm{COMM}(\Pi) \geq 5.$$

We remark that the result can be slightly improved by using the fact that there are additional rectangles corresponding to the sums 0, 1 and 2. However, that improvement is relatively small, and can be omitted for the sake of simplicity. ■

Additionally, there exists a trivial protocol for computing F, which we give as Protocol 9.

The total number of sent bits in Protocol 9 is obtained from the Steps 1 and 4. The length of characteristic vector in Step 1 is $2^n - 1$ as we can omit the bit for element 0 as it does not change the sum. The sum w in Step 4 can be represented with $2n - 1$ bits. Thus, the total number of transmitted bits is $2^n + 2n - 2$.

We obtain that for any deterministic protocol Π for computing (5.2), the communication complexity is bounded by

$$2^n + n - 1 \leq \mathrm{COMM}(\Pi) \leq 2^n + 2n - 2.$$

Compared to Example 16, we see that for this function the possible achieved gain over a trivial protocol which first synchronizes data is not significant. This suggests that asymptotic reduction in the number of transmitted bits cannot be achieved for some functions.

Protocol 9 Computing the sum of union of sets

Label
Π_9

Network topology
Nodes $\mathcal{V} = \{A, B\}$, edges $\mathcal{E} = \{(A,B),(B,A)\}$.

Randomness model
Deterministic

Input
Nodes A and B have sets $\mathcal{S}_A$ and $\mathcal{S}_B$.

Goal
Both nodes learn $F(\mathcal{S}_A, \mathcal{S}_B)$ for F defined in (5.2).

Online phase

$r = 1$

Transmission phase

1. Node A sends characteristic vector of $\mathcal{S}_A$ to node B.

$r = 2$

Computation phase

3. Node B computes $w = F(\mathcal{S}_A, \mathcal{S}_B)$.

Transmission phase

4. Node B sends w to node A.

5.2.2. Multiplication over integers

We now consider another function, a product of elements over the union of sets, namely the function

$$F(\mathcal{S}_A, \mathcal{S}_B) \triangleq \prod_{x \in \mathcal{S}_A \cup \mathcal{S}_B} x. \tag{5.6}$$

We present an analogous result for the function defined above. The lower bound on the communication complexity for computing F as in (5.6) is given as Theorem 29.

Theorem 29. *The number of bits communicated between the nodes A and B in any deterministic protocol* Π *that computes the function F in* (5.6) *is at least*

$$\mathrm{COMM}(\Pi) \geq 2^n + n - 2.$$

The proof of this theorem is similar to the proof of Theorem 28, but has significant nuances and thus we present it completely.

Proof. We obtain the bound by estimating the number of different F-monochromatic rectangles and then applying Lemma 27.

We denote by

$$\mathcal{J} \triangleq \mathbb{F}^n \setminus \{0,1\}.$$

In order to count the number of elements on the main diagonal, we define

$$\mathcal{B}_0 \triangleq \{(\mathcal{S}, \mathcal{J} \setminus \mathcal{S}) : \mathcal{S} \subseteq \mathcal{J}\}$$

$$= \{(\mathcal{S}_i, \mathcal{S}'_i) : i \in [2^{2^n-2}]\}.$$

Then, for every $(\mathcal{S}_i, \mathcal{S}'_i) \in \mathcal{B}_0$ we have:

$$F(\mathcal{S}_i, \mathcal{S}'_i) = \prod_{j=2}^{2^n-1} j = (2^n-1)!.$$

If we take $i, j \in [2^{n-2}]$ such that $i \neq j$, then we can consider two cases:

- If $\mathcal{S}_i \cup \mathcal{S}'_j \neq \mathcal{J}$, then there exists $x \in \mathcal{J}$, such that $x \notin \mathcal{S}_i \cup \mathcal{S}'_j$ and
$$F(\mathcal{S}_i, \mathcal{S}'_j) < (2^n-1)!.$$
- If $\mathcal{S}_i \cup \mathcal{S}'_j = \mathcal{J}$, since $\mathcal{S}_i \neq \mathcal{S}_j$, there exists $x \in \mathcal{S}_i \cap \mathcal{S}'_j$, thus $x \notin \mathcal{S}'_i \cup \mathcal{S}_j$ and
$$F(\mathcal{S}_j, \mathcal{S}'_i) < (2^n-1)!.$$

Thus, due to Lemma 26, there exists at least 2^{2^n-2} different F-monochromatic rectangles in $\mathcal{B}_0$.

For constructing additional F-monochromatic rectangles, we define

$$\mathcal{J}_\ell \triangleq \mathbb{F}^n \setminus \{0, 1, \ell\}$$

for every $\ell \in [2, 2^n-1]$ and

$$\begin{aligned}\mathcal{B}_\ell &\triangleq \{(\mathcal{S}, \mathcal{J}_\ell \setminus \mathcal{S}) : \mathcal{S} \subseteq \mathcal{J}_\ell\} \\ &= \{(\mathcal{S}_i, \mathcal{S}'_i) : i \in [2^{2^n-3}]\}.\end{aligned}$$

Then, for every pair $(\mathcal{S}_i, \mathcal{S}'_i) \in \mathcal{B}_\ell$ we have that

$$F(\mathcal{S}_i, \mathcal{S}'_i) = \prod_{\substack{j=2 \\ j\neq\ell}}^{2^n-1} j = \frac{(2^n-1)!}{\ell}.$$

By taking $i, j \in [2^{2^n-3}]$ such that $i \neq j$, similarly to the above cases we have either

$$F(\mathcal{S}_i, \mathcal{S}'_j) < \frac{(2^n-1)!}{\ell}$$

or

$$F(\mathcal{S}_j, \mathcal{S}'_i) < \frac{(2^n-1)!}{\ell}.$$

Due to Lemma 26, the set $\mathcal{B}_\ell$ contains 2^{2^n-3} F-monochromatic rectangles. As ℓ can be chosen in 2^n-2 ways, the total number of F-monochromatic rectangles in $\mathcal{B}_\ell$ over all $\ell \notin \{0,1\}$ is

$$(2^n-2) \cdot (2^{2^n-3}).$$

Additionally, we can count a single F-monochromatic rectangle corresponding to the value 0 of the function F. By adding all different F-monochromatic rectangles, we get that the total number of F-monochromatic rectangles is at least

$$\begin{aligned}\Upsilon(F) &\geq 2^{2^n-2} + (2^n-2)\cdot(2^{2^n-3}) + 1 \\ &= 2^{2^n+n-3} + 1.\end{aligned}$$

Due to Lemma 27, by rounding up to the next integer, the communication complexity of any deterministic protocol computing F defined in (5.6) is at least

$$\text{COMM}(\Pi) \geq 2^n + n - 2.$$

□

5.3. Reduction to known problems using Monte-Carlo style protocols

In this section, we consider randomized protocols for computing the sum function defined in (5.2). In order to obtain bounds on the communication complexity, we use a technique of reducing the problem of computing F to computing another function and measure the overhead.

5.3.1. Lower bounds using reduction to set disjointness problem

Given two sets $\mathcal{S}_A, \mathcal{S}_B \subseteq \mathbb{F}^n$, the binary set disjointness function DISJ is defined as:

$$\text{DISJ}(\mathcal{S}_A, \mathcal{S}_B) \triangleq \begin{cases} 1 & \text{if } \mathcal{S}_A \cap \mathcal{S}_B = \varnothing \\ 0 & \text{otherwise} \end{cases}.$$

In set disjointness problem, there are two nodes A and B possessing the sets $\mathcal{S}_A \subseteq \mathbb{F}^n$ and $\mathcal{S}_B \subseteq \mathbb{F}^n$, respectively. They want to cooperatively compute the value of the function $\text{DISJ}(\mathcal{S}_A, \mathcal{S}_B)$. The communication complexity of set disjointness problem has been well studied and variety of bounds has been established. For example, in [38] a lower bound of $2^n + 1$ was established using fooling set technique for any deterministic protocol. For randomized protocols, an asymptotically tight bound of $\Theta(2^n)$ has been proposed in [3], [29], [33] and [56].

Assuming that there exists a protocol Π_F for computing F, the following Protocol 10 allows to solve the set disjointness problem.

Lemma 30. *Protocol 10 computes the set disjointness function* $\text{DISJ}(\mathcal{S}_A, \mathcal{S}_B)$ *and the communication complexity of the protocol is upper bounded by*

$$\text{COMM}(\Pi_F) + 2n$$

bits, where protocol Π_F *computes the sum function* F *as given in* (5.2)*.*

Protocol 10 Computing set disjointness using Π_F as sub-protocol

Label

Π_{10}

Network topology

Nodes $\mathcal{V} = \{A, B\}$, edges $\mathcal{E} = \{(A,B),(B,A)\}$.

Randomness model

Randomness model of Π_F.

Input

Nodes A and B have sets $\mathcal{S}_A$ and $\mathcal{S}_B$.

Goal

Nodes obtain $\text{DISJ}(\mathcal{S}_A, \mathcal{S}_B)$.

Online phase

$r = 1$

Transmission phase

1. Node A sends bit 1 if $0 \in \mathcal{S}_A$ and 0 otherwise.

Recovery phase

2. If node B receives 1 and $0 \in \mathcal{S}_B$, $\text{DISJ}(\mathcal{S}_A, \mathcal{S}_B) = 0$ and halts.

$r = 2$

Computation phase

3. Node A computes
$$w_A = \sum_{x \in \mathcal{S}_A} x$$
and node B computes
$$w_B = \sum_{x \in \mathcal{S}_B} x.$$

Transmission phase

4. Node A sends w_A to node B.
5. Node A and node B run protocol Π_F to obtain $w = F(\mathcal{S}_A, \mathcal{S}_B)$.

Recovery phase

6. Node B checks if $w_A + w_B = w$. If true then $\text{DISJ}(\mathcal{S}_A, \mathcal{S}_B) = 1$. Otherwise $\text{DISJ}(\mathcal{S}_A, \mathcal{S}_B) = 0$.

Proof. We will first establish the correctness of the protocol. The protocol is correct, if the decisions taken in Steps 2 or 6 are correct.

Firstly, the decision taken in Step 2 is trivially correct as node B concludes that $\text{DISJ}(\mathcal{S}_A, \mathcal{S}_B) = 0$ only if $0 \in \mathcal{S}_A \cap \mathcal{S}_B$ and thus the sets $\mathcal{S}_A$ and $\mathcal{S}_B$ are not disjoint.

Secondly, as the elements in the sets $\mathcal{S}_A$ and $\mathcal{S}_B$ are non-negative integers, then the the sums of the elements is always monotonically increasing. Thus, we have that

$$w_A + w_B \geq w,$$

and the equality only holding if

$$\mathcal{S}_A \cap \mathcal{S}_B = \varnothing.$$

In this case, node B concludes that the sets are disjoint and outputs the correct decision.

In order to establish the communication complexity, we see that the total communication consists of sending a single bit in Step 1, running Π in Step 4 and sending w_A in Step 5. The length of the value w_A is up to $2n-1$ bits. Thus the total communication is

$$\begin{aligned} \text{COMM}(\Pi_{10}) &\leq 1 + \text{COMM}(\Pi_F) + 2n - 1 \\ &= \text{COMM}(\Pi_F) + 2n. \end{aligned} \tag{5.7}$$

□

By rewriting (5.7), we get the bound on protocol for computing the sum

$$\text{COMM}(\Pi_F) \geq \text{COMM}(\Pi_{10}) - 2n.$$

Due to previous discussion, we get a bound of

$$\text{COMM}(\Pi_F) \geq 2^n - 2n + 1$$

for any deterministic protocol and

$$\text{COMM}(\Pi_F) = \Omega(2^n)$$

for any randomized protocol.

In Section 5.2.1 we presented an upper bound of $\mathcal{O}(2^n)$ for any deterministic protocol computing the sum function. As any deterministic protocol can also be considered as a randomized protocol, then we get an asymptotically tight bound of $\Theta(2^n)$ for any randomized protocol computing the sum function.

5.3.2. Upper bound using reduction to finding the set intersection problem

We use another problem in computer science called finding the set intersection [9]. In finding set intersection problem, the nodes A and B possess the sets $\mathcal{S}_A$ and $\mathcal{S}_B$, respectively, and wish to find the intersection $\mathcal{S}_A \cap \mathcal{S}_B$. If

$$\max(|\mathcal{S}_A|, |\mathcal{S}_B|) = k,$$

then the following Theorem 31 gives an upper bound on the protocol for finding the set intersection.

Theorem 31 ([9, Theorem 3.1])**.** *There exists an $\mathcal{O}(\sqrt{k})$ round constructive randomized protocol for finding the set intersection problem with success probability $1-1/\mathtt{POLY}(k)$. In the model of shared randomness the total communication complexity is $\mathcal{O}(k)$ and in the model of private randomness it is $\mathcal{O}(k+\log(n))$.*

We denote the protocol for finding the set intersection as $\Pi_{\mathtt{INT}}$. Then, the following Protocol 11 describes a protocol for computing the sum function F using $\Pi_{\mathtt{INT}}$ as a sub-protocol.

Protocol 11 Computing F using $\Pi_{\mathtt{INT}}$ as sub-protocol

Label

Π_{11}

Network topology

Nodes $\mathcal{V}=\{A,B\}$, edges $\mathcal{E}=\{(A,B),(B,A)\}$.

Randomness model

Randomness model of $\Pi_{\mathtt{INT}}$

Input

Nodes A and B have sets $\mathcal{S}_A$ and $\mathcal{S}_B$.

Goal

Nodes obtain $F(\mathcal{S}_A,\mathcal{S}_B)$.

Online phase

For $r\in[\rho]$

Computation phase

1. Node A computes
$$w_A=\sum_{x\in\mathcal{S}_A}x$$
and node B computes
$$w_B=\sum_{x\in\mathcal{S}_B}x.$$

Transmission phase

2. Node A sends w_A to node B.
3. Node B sends w_B to node A.
4. Nodes A and B run $\Pi_{\mathtt{INT}}$ to find $\mathcal{S}_A\cap\mathcal{S}_B$.

Recovery phase

5. Nodes A and B recover
$$w\triangleq w_A+w_B-\sum_{x\in\mathcal{S}_A\cap\mathcal{S}_B}x. \tag{5.8}$$

Lemma 32. *Protocol 11 computes the sum function* (5.2) *with up to* $\mathrm{COMM}(\Pi_{INT})+4n-2$ *bits.*

Proof. The communication of Protocol 11 consists of transmissions in Steps 2, 3 and 4. We require up to $2n-1$ to represent values w_A and w_B. The communi-

cation complexity in Step 4 is $\mathrm{COMM}(\Pi_{\texttt{INT}})$ and thus the total communication complexity is

$$\mathrm{COMM}(\Pi_{11}) \leq \mathrm{COMM}(\Pi_{\texttt{INT}}) + 4n - 2.$$

Now, we need to ensure that $w = F(\mathcal{S}_A, \mathcal{S}_B)$. It is trivial to establish from (5.8) that:

$$\begin{aligned}
w &= w_A + w_B - \sum_{x \in \mathcal{S}_A \cap \mathcal{S}_B} x \\
&= \sum_{x \in \mathcal{S}_A} x + \sum_{x \in \mathcal{S}_B} x - \sum_{x \in \mathcal{S}_A \cap \mathcal{S}_B} x \\
&= \sum_{x \in \mathcal{S}_A \setminus \mathcal{S}_B} x + \sum_{x \in \mathcal{S}_A \cap \mathcal{S}_B} x + \sum_{x \in \mathcal{S}_B \setminus \mathcal{S}_A} x + \sum_{x \in \mathcal{S}_A \cap \mathcal{S}_B} x - \sum_{x \in \mathcal{S}_A \cap \mathcal{S}_B} x \\
&= \sum_{x \in \mathcal{S}_A \setminus \mathcal{S}_B} x + \sum_{x \in \mathcal{S}_A \cap \mathcal{S}_B} x + \sum_{x \in \mathcal{S}_B \setminus \mathcal{S}_A} x \\
&= \sum_{x \in \mathcal{S}_A \cup \mathcal{S}_B} x \\
&= F(\mathcal{S}_A, \mathcal{S}_B)
\end{aligned}$$

□

By applying the results of Theorem 31 and Lemma 32, we obtain that the communication complexity of the protocol for computing the sum is $\mathcal{O}(k) + 4n$ in the shared randomness model and $\mathcal{O}(k) + 4n + \mathcal{O}(\log(n))$ in private randomness model.

5.4. Las-Vegas style randomized protocol for computing sum function

In this section, we construct an error-free randomized protocol for computing the sum function as defined in (5.2).

For that, we require that there exists a set of hash functions $\mathcal{H}$, where every $H \in \mathcal{H}$ is a uniform hash function $H : \mathbb{F}^n \to \mathbb{F}^m$, i.e.

$$\forall H \in \mathcal{H}, \forall j \in \mathbb{F}^m : \Pr_{x \in \mathbb{F}^n}(H(x) = j) = 2^{-m}. \tag{5.9}$$

We also assume that the hash functions can be indexed, i.e. we can choose $H_i \in \mathcal{H}$ for any $i \in \mathbb{N}$.

We present the protocol for computing the error-free function as Protocol 12 and analyze its communication complexity in the subsequent Theorem 33.

Protocol 12 Error-free protocol for computing F

Label

Π_{12}

Network topology

Nodes $\mathcal{V} = \{A, B\}$, edges $\mathcal{E} = \{(A,B),(B,A)\}$.

Randomness model

Shared randomness

Input

Nodes A and B with sets $\mathcal{S}_A$ and $\mathcal{S}_B$.

Goal

Nodes obtain $F(\mathcal{S}_A, \mathcal{S}_B)$.

Online phase

For $r \in \mathbb{N}$

Computation phase

1. Node B computes the set

$$K_r \triangleq \{H_r(x) : x \in \mathcal{S}_B\}.$$

Transmission phase

2. Node B sends K_r to node A.

Recovery phase

3. Node A creates $L_r = \varnothing$.
4. For every $x \in \mathcal{S}_A$, node A adds x to L_r if $H_r(x) \notin K_r$.
5. If $|L_r| \neq d_A$ then go to Step 1.

$r+1$

Computation phase

6. Node A computes

$$w_A \triangleq \sum_{x \in L_\rho} x.$$

Transmission phase

7. Node A sends w_A to node B.

Recovery phase

8. Node B computes

$$w \triangleq w_A + \sum_{x \in \mathcal{S}_B} x.$$

$r+2$

Transmission phase

9. Node B sends w to node A.

Theorem 33. *Let the nodes A and B sample uniformly at random the sets $\mathcal{S}_A$ and $\mathcal{S}_B$, respectively. If $d_A \leq |\mathbb{F}^m|$, then Protocol 12 computes the sum function*

$F(\mathcal{S}_A, \mathcal{S}_B)$ as defined in (5.2) *without errors. Averaging over all sequences of hash functions $(H_i)_{i\in\mathbb{N}}$, where every H_i is chosen uniformly from the set of all hash functions $\mathcal{H}$ such that* (5.9) *holds, the protocol transmits on average*

$$mk_B \left(1 - \frac{2^{n-m}-1}{2^n-1}\right)^{-d_A} + 4n - 2$$

bits.

As mentioned in Section 5.1, we implicitly assume that the node A knows value d_A. If the value is not available to the node A, then the nodes have to run a protocol to estimate the value. This could be done by using the methods for estimating the size of symmetric set difference as described in Section 3.2.1.

Proof. We first show that if the protocol completes, both parties know the value $F(\mathcal{S}_A, \mathcal{S}_B)$. For that, we follow the accepting path in the protocol description.

Let ρ be the number of rounds when the loop in the Steps 1–5 of the protocol terminates. Then, as in Step 5, we have that $\left|L_\rho\right| = d_A$, then it implies that

$$L_\rho = \mathcal{S}_A \setminus \mathcal{S}_B$$

as otherwise it would mean that there exists $x \in L_\rho$ such that $x \in \mathcal{S}_B$. However, this is a contradiction as in Step 4 we check that $H_\rho(x) \notin K_\rho$ and thus also $x \notin \mathcal{S}_B$. The value w_A computed in Step 6 thus corresponds to

$$w_A = \sum_{x\in L_\rho} x = \sum_{x\in \mathcal{S}_A\setminus\mathcal{S}_B} x$$

and the value w computed by node B in Step 8 is

$$\begin{aligned} w &= w_A + \sum_{x\in\mathcal{S}_B} x \\ &= w_{x\in\mathcal{S}_A\setminus\mathcal{S}_B} x + \sum_{x\in\mathcal{S}_B} x \\ &= \sum_{x\in\mathcal{S}_A\cup\mathcal{S}_B} x \\ &= F(\mathcal{S}_A, \mathcal{S}_B). \end{aligned}$$

Thus, if the protocol terminates, both nodes learn the actual value of $F(\mathcal{S}_A, \mathcal{S}_B)$.

Secondly, we need to estimate the average number of transmitted bits over all possible numbers of rounds. For that, we see that the number of bits transmitted in Steps 2, 7 and 9 is, respectively

$$t_0 \triangleq mk_B, \tag{5.10}$$

$$t_1 \triangleq 2n-1,$$

$$t_2 \triangleq 2n-1. \tag{5.11}$$

In every protocol run, Steps 7 and 9 are invoked only once. However, Step 2 is run in total of ρ times. The jump from Step 5 to Step 1 is omitted when $\left|L_\rho\right| = d_A$. This happens only if there is no collision between elements in $\mathcal{S}_A \setminus \mathcal{S}_B$ for the chosen hash function H_ρ in the last round.

If (5.9) holds, then the probability of a collision for any $H \in \mathcal{H}$ and $y \in \mathbb{F}^n$ is

$$\begin{aligned}\Pr(\text{collision}) &= \Pr_{x \in \mathbb{F}^n}(H(x) = H(y) \mid x \neq y) \\ &= \frac{2^{n-m}-1}{2^n-1}.\end{aligned}$$

Thus, if $x \in \mathcal{S}_0$, then $H_\rho(x) \in K_\rho$. Otherwise, if $x \notin \mathcal{S}_A \setminus \mathcal{S}_0$, then $H_\rho(x) \notin K_\rho$ only if there is no collision with an element in K_ρ. The probability of this happening is

$$\begin{aligned}\Pr(\left|L_\rho\right| = d_A) &= \Pr(\text{no collision for every } x \in \mathcal{S}_A \setminus \mathcal{S}_0) \\ &= \left(1 - \frac{2^{n-m}-1}{2^n-1}\right)^{d_A}. \qquad (5.12)\end{aligned}$$

Next, we compute the number of communicated bits T_ρ during $\rho \in \mathbb{N}$ rounds. We denote

$$\begin{aligned}q_a &\triangleq \Pr(\text{accept}) = \Pr(\left|L_\rho\right| = d_A) \\ q_n &\triangleq \Pr(\text{not accept}) = 1 - q_a.\end{aligned}$$

Here, q_a is a probability that the protocol succeeds in computing the sum of all elements.

At first, we look at the cases where we limit the number of rounds to $\rho \in \{1,2,3\}$. To express the expected number of communicated bits in an instance of the protocol, which succeeds after at most ρ rounds, we use the random variable T_ρ. We have:

$$\begin{aligned}E(T_1) &= t_0 + t_1 + t_2, \\ E(T_2) &= q_a(t_0 + t_1 + t_2) + q_n(2t_0 + t_1 + t_2), \\ E(T_3) &= q_a(t_0 + t_1 + t_2) + q_n q_a(2t_0 + t_1 + t_2) + q_n q_n(3t_0 + t_1 + t_2).\end{aligned}$$

In general when bounding the number of rounds by ρ, the total number of expected number of communicated bits is:

$$E(T_\rho) = \sum_{i=0}^{\rho-1} {q_n}^i t_0 + t_1 + t_2.$$

By allowing unbounded number of rounds, we obtain

$$E(T_\infty) = \sum_{i=0}^{\infty} {q_n}^i t_0 + t_1 + t_2$$

$$
\begin{aligned}
&= t_0 \sum_{i=0}^{\infty} {q_n}^i + t_1 + t_2 \\
&= t_0 \frac{1}{1-q_n} + t_1 + t_2 \\
&= t_0 {q_a}^{-1} + t_1 + t_2.
\end{aligned}
$$

By replacing with (5.10)-(5.11) and (5.12), we get

$$
E(T_\infty) = mk_B \left(1 - \frac{2^{n-m}-1}{2^n-1}\right)^{-d_A} + 4n - 2,
$$

as claimed. □

In the protocol run, the sizes of vectors n, the size k_B of set $\mathcal{S}_B$ and difference size d_A are given. In order to minimize the number of communicated bits, we can choose the length of the hash function output m which minimizes the number of transmitted bits, i.e.

$$
\arg\min_m \left(mk_B \left(1 - \frac{2^{n-m}-1}{2^n-1}\right)^{-d_A} + 4n - 2 \right).
$$

In order for the protocol to be efficient, we assume that $m \ll n$. Under this assumption,

$$
\mathrm{COMM}(\Pi) = \arg\min_m \left(mk_B (1-2^m)^{-d_A} + 4n - 2 \right). \tag{5.13}
$$

There is no closed-form solution for (5.13). For giving an upper bound on the communication complexity, we can take

$$
m = \log_2 \left(\frac{d_A}{c} \right),
$$

for some constant c. Then

$$
mk_B (1-2^{-m})^{-d_A} + 4n - 2 \approx \log_2 \left(\frac{d_A}{c} \right) k_B \left(1 - \frac{c}{d_A}\right)^{-d_A} + 4n - 2. \tag{5.14}
$$

By grouping the values in the right-hand side of (5.14), we get an upper bound on $\mathrm{COMM}(\Pi)$ as

$$
\mathcal{O}(k_B \cdot \log(d_A) + n).
$$

5.5. Summary of results

We summarize the results from Chapter 5 in Table 12.

Table 12. Overview of protocols for computing sum over synchronized data.

Communication complexity	Randomness model	Method Reference
$\Theta(d \cdot n)$	Deterministic	Reduction to data synchronization Section 5.1.1
$\geq 2^n + n - 1$	Deterministic	Using F-monochromatic rectangles Section 5.2.1
$\leq 2^n + 2n - 2$	Deterministic	Exchange of characteristic vector Section 5.2.1
$\geq 2^n - 2n + 1$	Deterministic	Reduction to set disjointness using [38]. Section 5.3.1
$\Theta(2^n)$	Randomized	Reduction to set disjointness using [3, 29, 33, 56]. Section 5.3.1
$\mathcal{O}(k) + 4n$	Shared randomness	Reduction to set intersection problem using [9] Section 5.3.2
$\mathcal{O}(k) + 4n + \mathcal{O}(\log(n))$	Private randomness	Reduction to finding the intersection using [9] Section 5.3.2
$\mathcal{O}(k_B \cdot \log(d_A) + n)$	Shared randomness without errors	Exchanging hashed elements Section 5.4

6. CONCLUSION AND FUTURE WORK

6.1. Summary of contributions

In this thesis, we gave a high-level description of data distribution, data synchronization, and function computation problems. We **formulated a generic framework**, which allows us to describe the underlying network topology as a graph and the protocols for all problems using unified notation. We have **identified criteria for data exchange scenarios** and **compiled a list of 14 relevant scenarios** in Table 1.

For data synchronization protocols, we used IBFs which allow us to achieve theoretically optimal communication complexity. However, the theoretical result relies on the knowledge of the number of elements in the symmetric set difference as otherwise, the behavior of IBF is unknown. Even though a protocol using Strata Estimator exists for estimating the size of the symmetric difference, it incurs additional overhead on top of synchronization. We looked at the method for analyzing IBFs to understand if it is possible to study the behavior of IBF in case the number of elements exceeds a threshold and **concluded that the current method is insufficient**.

We described an alternative representation for an IBF as a state matrix. Using this representation, we precisely **characterized when an IBF allows to extract only a subset of inserted elements**. We presented the result as Lemma 10. Using this Lemma, we **formulated Theorem 11 which complements existing results on IBFs**, which we have formulated as Theorems 2 and 12.

The new result allowed us to **describe an iterative data synchronization protocol**, which we gave as Protocol 5. As the new protocol enables partial or complete failures in every round, the nodes succeed in synchronizing their sets even if their estimate on the size of the symmetric difference is inexact or unknown. We **estimated theoretically the number of rounds required for complete data synchronization** and **compared the theoretical results to experimental simulations**. We observed that the simulations follow the theoretical estimates closely.

For data distribution protocols, we relied on existing results from index coding and data exchange problems. As those problems are defined only on a star and complete graphs, we required additional notation for extending the results to arbitrary networks. We **identified a sufficient condition of a network topology which ensures that for any assignment of has- and request-sets there exists a protocol satisfying every nodes' request**. We defined the network topology to be ρ-solvable if the protocol requires not more than ρ rounds. Additionally, we observed that the condition is related to the diameter of the underlying graph of the network topology.

For 1-solvable networks, we **gave a complete description of a protocol which satisfies the requests' of all nodes**. Additionally, **we proved that the described protocol is optimal**, i.e. every other protocol satisfying all requests transmits at

least the same number of bits as the given protocol. The protocol description is given as Protocol 6 and the optimality result is given as Theorem 18.

For arbitrary ρ-solvable networks, with $\rho \neq 1$, we **proposed a lower bound on the communication complexity** for any protocol satisfying the requests in Lemma 21. For protocol analysis, **we introduced new algebra which allows to study data dissemination in the network using matrix operations**. The relation between matrix operations and data dissemination was stated and proved in Lemma 22 and Corollary 23. We **described a protocol for data distribution in arbitrary network** as Protocol 7 and **proved its correctness** in Theorem 24.

We **defined a new problem called function computation on synchronized data**. Even as the problem generalizes data synchronization, we **showed through a counter-example that tools for data synchronization do not allow to achieve optimal communication complexity**. The counter-example was given as Example 16.

We **used F-monochromatic rectangles to obtain bounds on the communication complexity for sum and product functions** in deterministic randomness model. The results are given as Theorems 28 and 29. We also considered protocols where the nodes have access to randomness. Using reduction from set disjointness function, we **established a lower bound on the communication complexity for computing sum function** as Lemma 30. Using reduction to set intersection function, we **obtained an upper bound on the communication complexity for computing the sum function** as Lemma 32.

The previous reductions have constant complexity, but may occasionally output incorrect value. We **described a randomized error-free protocol for computing the sum function**. We proved the correctness of the protocol and analyzed the estimated communication complexity. The protocol description is given as Protocol 12 and the communication complexity as Theorem 33.

6.1.1. Proposed data exchange scenarios

In Section 2.6.2, we compiled a list of scenarios for data exchange. To indicate the contribution of this thesis, we can consider which scenarios we have a positive result. We give the overview in the following Table 13. Novel protocols proposed in this thesis are indicated in bold.

Not covered in this dissertation, but we have discussed in [36] that Protocol 5 can be used in scenarios 3 and 5. For these scenarios, we have denoted the protocol in cursive.

6.2. Future directions

In addition to the results obtained in this dissertation, we anticipate novel related research questions.

Table 13. Overview of positive results in this thesis

scenario	graph type	trans. model	data exchange	protocol
1	pair	unicast	synchronization	Protocols 4, **5**
2	complete	broadcast	distribution	Protocol 3
3	complete	broadcast	synchronization	Protocol *5*
4	star	broadcast	distribution	Protocols 2, **6**
5	star	broadcast	synchronization	Protocol *5*
6	strongly connected	unicast	distribution	-
7	strongly connected	unicast	synchronization	-
8	strongly connected	broadcast	distribution	Protocol **7**
9	strongly connected	broadcast	synchronization	-
10	pair	unicast	function comp.	Protocols **11, 12**
11	complete	broadcast	function comp.	-
12	star	broadcast	function comp.	-
13	strongly connected	unicast	function comp.	-
14	strongly connected	broadcast	function comp.	-

6.2.1. Data synchronization in arbitrary network topologies

Both existing and our proposed protocols for data synchronization are defined only for networks of two nodes. Mitzenmacher and Pagh proposed in [50] an extension to Protcol 4 for network topologies where the underlying graph is a weakly connected graph utilizing methods from network coding.

Their proposal consists of two observations. First, for complete graphs, by encoding IBFs over extension field of $\mathbb{F}_p$, with $p \geq u$, where u is the number of nodes, relevant IBF procedures work exactly as for binary fields. When adding IBFs from all nodes using $\mathsf{Add}()$ procedure, the procedure $\mathsf{Extract}()$ extracts the set

$$\cup_{v \in \mathcal{V}} \mathcal{S}_v \setminus \cap_{v \in \mathcal{V}} \mathcal{S}_v.$$

Thus, every node can recover the missing elements from their sets.

Their second observation is that in weakly connected graphs it is possible to construct a linear combination of several IBFs. Thus, it is possible to apply network coding methods to encode packets transmitted in the network.

Their approach however requires that the IBF has enough cells so that the extraction succeeds with high probability. However, as estimating the size of the symmetric difference of two sets is a difficult problem, then estimating

$$|\cup_{v \in \mathcal{V}} \mathcal{S}_v \setminus \cap_{v \in \mathcal{V}} \mathcal{S}_v|$$

in non-complete graphs is orders of magnitude more difficult. We believe that our proposed iterative protocol combined with network coding would yield a practical data synchronization protocol for arbitrary networks. A positive result allows us to give a protocol for data exchange scenario 9.

Using results for network coding in unicast networks [41,45,61] may allow us to describe data synchronization in scenario 7.

6.2.2. Function computation in arbitrary network topologies

One of the techniques used in Chapter 5 used F-monochromatic rectangles to upper bound the communication complexity of computing a function. It is shown in [38] that it is possible to generalize the concept into u-dimensional space and use F-monochromatic *cylinders*. Even though the method itself generalizes, we have not yet considered the specific extension to larger dimensions. A positive result would allow us to provide a protocol for scenario 11.

BIBLIOGRAPHY

[1] Rudolf Ahlswede, Ning Cai, Shuo-Yen R. Li, and Raymond W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, 2000.

[2] Ziv Bar-Yossef, Yitzhak Birk, Thathachar S. Jayram, and Tomer Kol. Index coding with side information. *IEEE Transactions on Information Theory*, 57(3):1479–1494, March 2011.

[3] Ziv Bar-Yossef, Thathachar S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences*, 68(4):702–732, 2004.

[4] Steven J. Benson, Yinyu Ye, and Xiong Zhang. Solving large-scale sparse semidefinite programs for combinatorial optimization. *SIAM Journal on Optimization*, 10(2):443–461, 2000.

[5] Yitzhak Birk and Tomer Kol. Coding on demand by an informed source (ISCOD) for efficient broadcast of different supplemental data to caching clients. *IEEE Transactions on Information Theory*, 52(6):2825–2830, 2006.

[6] Anudhyan Boral and Michael Mitzenmacher. Multi-party set reconciliation using characteristic polynomials. In *52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton 2014)*, pages 1182–1187, 2014.

[7] Andrei Z. Broder. On the resemblance and containment of documents. In *Compression and Complexity of Sequences (SEQUENCES'97)*, pages 21–29. IEEE Computer Society, 1997.

[8] Andrei Z. Broder, Moses Charikar, Alan M. Frieze, and Michael Mitzenmacher. Min-wise independent permutations. *Journal of Computer and System Sciences*, 60(3):630–659, 2000.

[9] Joshua Brody, Amit Chakrabarti, Ranganath Kondapally, David P. Woodruff, and Grigory Yaroslavtsev. Beyond set disjointness: the communication complexity of finding the intersection. In *ACM Symposium on Principles of Distributed Computing (PODC 14)*, pages 106–113, 2014.

[10] Mohammad Asad R. Chaudhry and Alex Sprintson. Efficient algorithms for index coding. In *IEEE INFOCOM Workshops 2008*, pages 1–4, 2008.

[11] Graham Cormode and Shan Muthukrishnan. What's new: finding significant differences in network data streams. *IEEE/ACM Transactions on Networking*, 13(6):1219–1232, 2005.

[12] Thomas A. Courtade and Richard D. Wesel. Coded cooperative data exchange in multihop networks. *IEEE Transactions on Information Theory*, 60(2):1136–1158, 2014.

[13] Mayur Datar and Shan Muthukrishnan. Estimating rarity and similarity over data stream windows. In *European Symposium on Algorithms 2002 (ESA 2002)*, pages 323–335, 2002.

[14] Son Hoang Dau, Vitaly Skachek, and Yeow Meng Chee. On the security of index coding with side information. *IEEE Transactions on Information Theory*, 58(6):3975–3988, 2012.

[15] Son Hoang Dau, Vitaly Skachek, and Yeow Meng Chee. Error correction for index coding with side information. *IEEE Transactions on Information Theory*, 59(3):1517–1531, 2013.

[16] Martin Dietzfelbinger, Andreas Goerdt, Michael Mitzenmacher, Andrea Montanari, Rasmus Pagh, and Michael Rink. Tight thresholds for Cuckoo Hashing via XORSAT. In *International Colloquium on Automata, Languages, and Programming*, pages 213–225. Springer, 2010.

[17] Martin Dietzfelbinger and Stefan Walzer. Dense peelable random uniform hypergraphs. In *27th Annual European Symposium on Algorithms (ESA 2019)*, 2019.

[18] Andrew Duncan. Powers of the adjacency matrix and the walk matrix. 2004.

[19] Michelle Effros, Salim El Rouayheb, and Michael Langberg. An equivalence between network coding and index coding. *IEEE Transactions on Information Theory*, 61(5):2478–2487, 2015.

[20] Salim El Rouayheb, Alex Sprintson, and Costas Georghiades. On the index coding problem and its relation to network coding and matroid theory. *IEEE Transactions on Information Theory*, 56(7):3187–3195, 2010.

[21] Salim El Rouayheb, Alex Sprintson, and Parastoo Sadeghi. On coding for cooperative data exchange. In *2010 IEEE Information Theory Workshop on Information Theory (ITW 2010, Cairo)*, pages 1–5. IEEE, 2010.

[22] David Eppstein, Michael T. Goodrich, Frank Uyeda, and George Varghese. What's the difference?: Efficient set reconciliation without prior context. *ACM SIGCOMM Computer Communication Review*, 41(4):218–229, August 2011.

[23] Maryam Fazel. *Matrix rank minimization with applications*. PhD thesis, PhD thesis, Stanford University, 2002.

[24] Robert G. Gallager. *Low-density parity-check codes*. PhD thesis, MIT, 1963.

[25] James F. Geelen. Maximum rank matrix completion. *Linear Algebra and its Applications*, 288:211–217, 1999.

[26] Marco Gentili. Set reconciliation and file synchronization using invertible bloom lookup tables. Master's thesis, Harvard College, 2015.

[27] Mira Gonen and Michael Langberg. Coded cooperative data exchange problem for general topologies. *IEEE Transactions on Information Theory*, 61(10):5656–5669, 2015.

[28] Michael T. Goodrich and Michael Mitzenmacher. Invertible Bloom lookup tables. In *49th Annual Allerton Conference on Communication, Control, and Computing (Allerton 2011)*, pages 792–799. IEEE, 2011.

[29] Johan Håstad and Avi Wigderson. The randomized communication complexity of set disjointness. *Theory of Computing*, 3(1):211–219, 2007.

[30] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *30th Annual ACM Symposium on Theory of Computing (STOC'98)*, pages 604–613. Association for Computing Machinery, 1998.

[31] Syed Ali Jafar. Topological interference management through index coding. *IEEE Transactions on Information Theory*, 60(1):529–568, 2014.

[32] Mingyue Ji, Antonia M. Tulino, Jaime Llorca, and Giuseppe Caire. Caching and coded multicasting: Multiple groupcast index coding. In *2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 881–885, 2014.

[33] Bala Kalyanasundaram and Georg Schintger. The probabilistic communication complexity of set intersection. *SIAM Journal on Discrete Mathematics*, 5(4):545–557, 1992.

[34] Jae-Won Kim and Jong-Seon No. Index coding with erroneous side information. *IEEE Transactions on Information Theory*, 63(12):7687–7697, 2017.

[35] Ivo Kubjas and Vitaly Skachek. Data dissemination problem in wireless networks. In *53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton 2015)*, pages 1197–1204, 2015.

[36] Ivo Kubjas and Vitaly Skachek. Failure probability analysis for partial extraction from Invertible Bloom Filters, 2020.

[37] Eyal Kushilevitz. Communication complexity. *Advances in Computers*, 44:331–360, 1997.

[38] Eyal Kushilevitz and Noam Nisan. *Communication complexity*. Cambridge University Press, 1997.

[39] Leslie Lamport. *The Part-Time Parliament*, page 277–317. Association for Computing Machinery, New York, NY, USA, 2019.

[40] Michael Langberg and Alex Sprintson. On the hardness of approximating the network coding capacity. In *2008 IEEE International Symposium on Information Theory*, pages 315–319, 2008.

[41] Zongpeng Li and Baochun Li. Network coding: The case of multiple unicast sessions. In *52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton 2004)*, volume 16, 2004.

[42] Yucheng Liu, Parastoo Sadeghi, Fatemeh Arbabjolfaei, and Young-Han Kim. On the capacity for distributed index coding. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 3055–3059, 2017.

[43] Yuxin Liu, Badri N. Vellambi, Young-Han Kim, and Parastoo Sadeghi. On the capacity region for secure index coding. In *2018 IEEE Information Theory Workshop (ITW)*, pages 1–5, 2018.

[44] Michael G. Luby, Michael Mitzenmacher, Mohammad Amin Shokrollahi, and Daniel A. Spielman. Efficient erasure correcting codes. *IEEE Transactions on Information Theory*, 47(2):569–584, 2001.

[45] Desmond S. Lun, Muriel Médard, and Ralf Koetter. Network coding for efficient wireless unicast. In *2006 International Zurich Seminar on Communications*, pages 74–77. IEEE, 2006.

[46] Shiqian Ma, Donald Goldfarb, and Lifeng Chen. Fixed point and Bregman iterative methods for matrix rank minimization. *Mathematical Programming*, 128(1-2):321–353, 2011.

[47] Hamed Maleki, Viveck R. Cadambe, and Syed A. Jafar. Index coding - an interference alignment perspective. *IEEE Transactions on Information Theory*, 60(9):5402–5432, 2014.

[48] Yaron Minsky and Ari Trachtenberg. Practical set reconciliation. In *40th Annual Allerton Conference on Communication, Control, and Computing*, volume 248. Citeseer, 2002.

[49] Yaron Minsky, Ari Trachtenberg, and Richard Zippel. Set reconciliation with nearly optimal communication complexity. *IEEE Transactions on Information Theory*, 49(9):2213–2218, 2003.

[50] Michael Mitzenmacher and Rasmus Pagh. Simple multi-party set reconciliation. *Distributed Computing*, 31(6):441–453, 2018.

[51] Michael Molloy. The pure literal rule threshold and cores in random hypergraphs. In *15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA04)*, pages 672–681, 2004.

[52] Shanmugavelayutham Muthukrishnan. *Data streams: Algorithms and applications*. Now Publishers Inc, 2005.

[53] National Institute of Standards and Technology. Federal Information Processing Standards Publication 180-4: Secure Hash Standard (SHS)., 2015.

[54] Lawrence Ong, Chin Keong Ho, and Fabian Lim. The single-uniprior index-coding problem: The single-sender case and the multi-sender extension. *IEEE Transactions on Information Theory*, 62(6):3165–3182, 2016.

[55] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In *2014 USENIX Annual Technical Conference (USENIX ATC 14)*, pages 305–319, Philadelphia, PA, June 2014. USENIX Association.

[56] Alexander A. Razborov. On the distributional complexity of disjointness. In Michael S. Paterson, editor, *Automata, Languages and Programming*, pages 249–253, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg.

[57] Thomas J. Richardson and Rüdiger L. Urbanke. Efficient encoding of low-density parity-check codes. *IEEE Transactions on Information Theory*, 47(2):638–656, 2001.

[58] Michael Rink. Mixed hypergraphs for linear-time construction of denser hashing-based data structures. In Peter van Emde Boas, Frans C. A. Groen, Giuseppe F. Italiano, Jerzy Nawrocki, and Harald Sack, editors, *SOFSEM 2013: Theory and Practice of Computer Science*, pages 356–368, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[59] Parastoo Sadeghi, Fatemeh Arbabjolfaei, and Young-Han Kim. Distributed index coding. In *2016 IEEE Information Theory Workshop (ITW)*, pages 330–334, 2016.

[60] Robert Schweller, Zhichun Li, Yan Chen, Yan Gao, Ashish Gupta, Yin Zhang, Peter A. Dinda, Ming-Yang Kao, and Gokhan Memik. Reversible sketches: Enabling monitoring and analysis over high-speed data streams. *IEEE/ACM Transactions on Networking*, 15(5):1059–1072, 2007.

[61] Sudipta Sengupta, Shravan Rayanchu, and Suman Banerjee. An analysis of wireless network coding for unicast sessions: The case for coding-aware routing. In *26th IEEE International Conference on Computer Communications (IEEE INFOCOM 2007)*, pages 1028–1036. IEEE, 2007.

[62] Vitaly Skachek and Michael G. Rabbat. Subspace synchronization: A network-coding approach to object reconciliation. In *2014 IEEE International Symposium on Information Theory*, pages 2301–2305, 2014.

[63] Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In *11h Annual ACM Symposium on Theory of Computing (STOC '79)*, STOC '79, pages 209–213, 1979.

[64] Daichi Yugawa and Tadashi Wadayama. Finite length analysis on listing failure probability of invertible bloom lookup tables. *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, 97(12):2309–2316, January 2014.

ACKNOWLEDGEMENT

After a semester in the Cyber Security Master's curriculum, I realized that I want to return to Tartu for a more theoretical perspective. Unofficially, it was February 25th in 2013 when I started my PhD by writing the following letter to Vitaly:

> "I am taking the research seminar in cryptology. The set reconciliation with interpolation seems interesting and I have read the referred paper. If I would choose that subject then what would be the expected outcome? Just a report or could there be any improvements and some proof-of-concept implementation?"

After the report, Vitaly suggested working further on the topic, leading me to write a Master's thesis and my first publication. He invited me to continue the research as a PhD student, which eventually led to completing the current dissertation. Without his suggestions, I would never have started doing my PhD, and for that, I am genuinely grateful.

But the studies is not only about doing research. Students are expected to present the results at conferences and be abroad for research cooperation, teach and take courses, and do administrative tasks. Without the help of Natali Belinska, Ülle Holm, Martin Kaljula, Heili Kase, Heisi Kurig, Reili Liiver, Liivi Luik, Eva Pruusapuu and Anneli Vainumäe, I wouldn't have managed through all the bureaucracy. Thank you for all the help. I also want to thank the rest of the institute for creating a productive and cooperative environment.

This dissertation was reviewed by Professor Parastoo Sadeghi, Professor Tadashi Wadayama and Associate Professor Dirk Oliver Theis. I humbly thank you for your suggestions to improve the content. These comments have allowed me to clarify the thesis and make it more readable.

I want to thank my fellow students and co-authors for all the exciting discussions on research and life. Healthy debate indeed allows us to see different aspects of the problems and be critical about the ideas.

My parents and my sister have been motivating examples of working hard towards the goal. They have always encouraged me to move forwards with my studies and occasionally taking time off. Thank you.

My most tremendous gratitude goes to my lovely wife Leili and my sweetest daughter Ella Mei. You kept me on track and pushed me through my hesitations during the isolation in the pandemic. I wouldn't have finished without your support.

My research was supported by the grants PRG49, PUT405, IUT2-1 from the Estonian Research Council, the European Regional Development Fund via CoE project EXCITE, and the grant EMP133 from the Norwegian-Estonian Research Cooperation Programme. My research visit to Aalto University was funded by Dora Plus Doctoral Student mobility grant from European Regional Development Fund.

SISUKOKKUVÕTE

Algebralised lahendused detsentraliseeritud süsteemides tekkivatele probleemidele

Seoses järjest suureneva pilvetehnoloogia platformide ja uute juhtmeta tehnoloogiate kasutamisega on terminalide vahelise suhtluse põhimõtted muutunud serverklient mudelist keerulisteks detsentraliseeritud mudeliteks. Teenusepakkujatel on tarvis hajutada oma teenuseid erinevate andmekeskuste vahel, et olla võimeline töötlema suurt päringute mahtu ja olla madala latentsuse jaoks klientidele füüsiliselt lähedal. Kuid andmete täielik dubleerimine mitmete serverite vahel on ressursse raiskav ja rahaliselt kulukas.

Käesolevas doktoritöös uurime me kolme suunda, mis võimaldavad vähendada serverite ja kasutajate vahel edastatud andmete mahtu. Kõigis kolmes suunas esitame me vastava probleemi matemaatilise struktuurina ja rakendame meetodeid algebrast lahendamaks antud probleemi.

Esimese probleemina uurime me andmete sünkroniseerimist. Andmete sünkroniseerimise probleemis on terminalid sõltumatute andmehulkadega ja nende eesmärgiks on leida oma andmehulkade ühend. Lihtne lahendus, mis seisneb hulkade (või isegi ainult vastavate elementide indeksite) täielikus vahetamises, muutub kiiresti ebaefektiivseks kui hulkade kardinaalsused muutuvad suureks. Varasemalt on välja pakutud meetod kasutades pööratavaid Bloomi filtreid (invertible Bloom filter - IBF), mida kasutades on vahetatavate andmete hulk asümptootiliselt samaväärne elementide arvuga hulkade sümmeetrilises vahes. Antud protokolli kasutamiseks on tarvilik teada, mitu elementi on iga terminali andmehulgas puudu, kuid selle suuruse teadasaamine on andmevahetuse mõttes kulukas ja lisab olulise ülekulu andmete sünkroniseerimise protokollile. Osas 3 esitame me IBFi esituse maatrikskujul ja kombinatoorse argumendi hindamaks osalise lahti pakkimise tõrke tõenäosust. Me nimetame uut esitust *osaliselt lahti pakitavaks IBFiks*. Lisaks kirjeldame me iteratiivset andmete sünkroniseerimise protokolli, mis lubab leida andmehulkade ühendit juhul kui hulkade sümmeetrilise vahe suurus on teadmata või ebatäpne. Täiendavalt võrdleme me teoreetilisi tulemusi eksperimentaalsete simulatsioonidega.

Teise suunana uurime me andmete jaotamist. Andmete jaotamise probleemis võib võrgutopoloogiat kirjeldav suunatud graaf olla suvaline tugevalt sidus graaf. Terminalide eesmärgiks on teada saada teistelt terminalidelt päritud elemendid. Osas 4 formuleerime me uue tingimuse, mis kirjeldab minimaalset seeriate hulka igas lahenduvas protokollis. Me nimetame seda tingimust r-lahenduvuseks. Me kasutame tehnikaid indeks-koodidest ja andmete edastamise probleemidest konstrueerimaks protokolle iga r-lahenduva võrgu jaoks. Vaadates 1-lahenduvaid võrke erijuhtumina, pakume me välja optimaalse protokolli, mis edastab minimaalse hulga bitte üle kõikvõimalike protokollide. Suvalise r-lahenduva võrgu korral on välja pakutud protokollid minimaalsete seeriate arvuga.

Viimasena uurime me osas 5 funktsioonide arvutamist sünkroniseeritud andmetel. See probleem erineb andmete sünkroniseerimise probleemist, kuna terminalide eesmärgiks on teada saada ainult funktsioon terminalide andmehulkade ühendist. Me näeme, et probleemi definitsiooni muutus lubab meil oluliselt vähendada edastatud andmete mahtu. Me pakume välja ülemise tõkke andmeedastuse keerukusele teatavatel funktsioonide perekonnal kasutades F-monokromaatilisi ristkülikuid. Lisaks kasutame me taandamist hulkade ühisosa ja lõikumatute hulkade probleemidele, et arvutada asümptootilist andmevahetuse keerukust. Me kirjeldame ja uurime veavabat protokolli funktsiooni väärtuse arvutamiseks sünkroniseeritud andmetel kasutades räsifunktsioonide perekondi.

CURRICULUM VITAE

Personal data

Name:	Ivo Kubjas
Date of birth:	September 20th, 1989
Place of birth:	Tartu, Estonia
Citizenship:	Estonia
Languages:	Estonian and English
Contact:	ivo.kubjas@eesti.ee

Education

2014 -	University of Tartu PhD candidate in Computer Science
2012 - 2014	University of Tartu and Tallinn University of Technology MSc in Cyber Security
2008 - 2012	University of Tartu BSc in Mathematics

Employment

2017	Aalto ülikool, Visiting Doctoral Student
2014 -	Smartmatic-Cybernetica Centre of Excellence for Internet Voting OÜ, Software Engineer
2014	University of Tartu, Specialist
2013 -	Teokon OÜ, Founder
2012-2013	OÜ Ideelabor, Software Developer

Scientific work

Main fields of interest:

- Decentralized systems
- Online voting

ELULOOKIRJELDUS

Isikuandmed

Nimi:	Ivo Kubjas
Sünniaeg:	September 20, 1989
Sünnikoht:	Tartu, Eesti
Kodakondsus:	Eesti
Keeleoskus:	eesti ja inglise
Kontakt:	ivo.kubjas@eesti.ee

Haridus

2014 -	Tartu Ülikool arvutiteaduste doktorant
2012 - 2014	Tartu Ülikool ja Tallinna Tehnikaülikool küberkaitse magister
2008 - 2012	Tartu Ülikool matemaatika bakalaureus

Teenistuskäik

2017	Aalto ülikool, külalisdoktorant
2014 -	Smartmatic-Cybernetica Centre of Excellence for Internet Voting OÜ, tarkvara inserner
2014	Tartu Ülikool, spetsialist
2013 -	Teokon OÜ, asutaja
2012-2013	OÜ Ideelabor, tarkvara arendaja

Teadustegevus

Peamised uurimisvaldkonnad:

- detsentraliseeritud süsteemid
- internetihääletamine

LIST OF ORIGINAL PUBLICATIONS

The following publications by the author serve the basis of this dissertation:

1. Ivo Kubjas and Vitaly Skachek. Data dissemination problem in wireless networks. In *Proc. 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1197-1204, Oct. 2015.
2. Ivo Kubjas and Vitaly Skachek. Two-party function computation on the reconciled data. In *Proc. 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 390-396, Oct. 2017.
3. Ivo Kubjas and Vitaly Skachek. Failure Probability Analysis for Partial Extraction from Invertible Bloom Filters. *arXiv preprint arXiv:2008.00879, 2020*. (Submitted for publication in IEEE Transactions on Information Theory)

Additionally, the author has co-authored the following articles during PhD studies which are not included in this dissertation:

4. Ragnar Freij-Hollanti, Oliver Gnilke, Camilla Hollanti, Anna-Lena Horlemann-Trautmann, David Karpuk, and Ivo Kubjas. Reed-Muller Codes for Private Information Retrieval. *International Workshop on Coding and Cryptography (WCC 2017)*, Sept. 2017.
5. Ivo Kubjas, Tiit Pikma, and Jan Willemson. Estonian Voting Verification Mechanism Revisited Again. In *Springer Proceedings of the Second Joint International Conference on Electronic Voting (E-Vote-ID 2017)*, pages 306-317, Oct. 2017.
6. Kristjan Krips, Ivo Kubjas, and Jan Willemson. An Internet Voting Protocol with Distributed Verification Receipt Generation. In *TUT Press Proceedings of the Third Joint International Conference on Electronic Voting (E-Vote-ID 2018)*, pages 128-146, Oct. 2018.
7. Sven Heiberg, Ivo Kubjas, Janno Siim, and Jan Willemson. On Trade-offs of Applying Block Chains for Electronic Voting Bulletin Boards. In *TUT Press Proceedings of the Third Joint International Conference on Electronic Voting (E-Vote-ID 2018)*, pages 259-276, Oct. 2018.
8. Ragnar Freij-Hollanti, Oliver Gnilke, Camilla Hollanti, Anna-Lena Horlemann-Trautmann, David Karpuk, and Ivo Kubjas. t-Private Information Retrieval Schemes Using Transitive Codes. *IEEE Transactions on Information Theory*, volume 65, issue 4, pages 2107-2118, April 2019.

DISSERTATIONES INFORMATICAE PREVIOUSLY PUBLISHED IN DISSERTATIONES MATHEMATICAE UNIVERSITATIS TARTUENSIS

19. **Helger Lipmaa.** Secure and efficient time-stamping systems. Tartu, 1999, 56 p.
22. **Kaili Müürisep.** Eesti keele arvutigrammatika: süntaks. Tartu, 2000, 107 lk.
23. **Varmo Vene.** Categorical programming with inductive and coinductive types. Tartu, 2000, 116 p.
24. **Olga Sokratova.** Ω-rings, their flat and projective acts with some applications. Tartu, 2000, 120 p.
27. **Tiina Puolakainen.** Eesti keele arvutigrammatika: morfoloogiline ühestamine. Tartu, 2001, 138 lk.
29. **Jan Villemson.** Size-efficient interval time stamps. Tartu, 2002, 82 p.
45. **Kristo Heero.** Path planning and learning strategies for mobile robots in dynamic partially unknown environments. Tartu 2006, 123 p.
49. **Härmel Nestra.** Iteratively defined transfinite trace semantics and program slicing with respect to them. Tartu 2006, 116 p.
53. **Marina Issakova.** Solving of linear equations, linear inequalities and systems of linear equations in interactive learning environment. Tartu 2007, 170 p.
55. **Kaarel Kaljurand.** Attempto controlled English as a Semantic Web language. Tartu 2007, 162 p.
56. **Mart Anton.** Mechanical modeling of IPMC actuators at large deformations. Tartu 2008, 123 p.
59. **Reimo Palm.** Numerical Comparison of Regularization Algorithms for Solving Ill-Posed Problems. Tartu 2010, 105 p.
61. **Jüri Reimand.** Functional analysis of gene lists, networks and regulatory systems. Tartu 2010, 153 p.
62. **Ahti Peder.** Superpositional Graphs and Finding the Description of Structure by Counting Method. Tartu 2010, 87 p.
64. **Vesal Vojdani.** Static Data Race Analysis of Heap-Manipulating C Programs. Tartu 2010, 137 p.
66. **Mark Fišel.** Optimizing Statistical Machine Translation via Input Modification. Tartu 2011, 104 p.
67. **Margus Niitsoo**. Black-box Oracle Separation Techniques with Applications in Time-stamping. Tartu 2011, 174 p.
71. **Siim Karus.** Maintainability of XML Transformations. Tartu 2011, 142 p.
72. **Margus Treumuth.** A Framework for Asynchronous Dialogue Systems: Concepts, Issues and Design Aspects. Tartu 2011, 95 p.
73. **Dmitri Lepp.** Solving simplification problems in the domain of exponents, monomials and polynomials in interactive learning environment T-algebra. Tartu 2011, 202 p.

74. **Meelis Kull.** Statistical enrichment analysis in algorithms for studying gene regulation. Tartu 2011, 151 p.
77. **Bingsheng Zhang.** Efficient cryptographic protocols for secure and private remote databases. Tartu 2011, 206 p.
78. **Reina Uba.** Merging business process models. Tartu 2011, 166 p.
79. **Uuno Puus.** Structural performance as a success factor in software development projects – Estonian experience. Tartu 2012, 106 p.
81. **Georg Singer.** Web search engines and complex information needs. Tartu 2012, 218 p.
83. **Dan Bogdanov.** Sharemind: programmable secure computations with practical applications. Tartu 2013, 191 p.
84. **Jevgeni Kabanov.** Towards a more productive Java EE ecosystem. Tartu 2013, 151 p.
87. **Margus Freudenthal.** Simpl: A toolkit for Domain-Specific Language development in enterprise information systems. Tartu, 2013, 151 p.
90. **Raivo Kolde.** Methods for re-using public gene expression data. Tartu, 2014, 121 p.
91. **Vladimir Šor.** Statistical Approach for Memory Leak Detection in Java Applications. Tartu, 2014, 155 p.
92. **Naved Ahmed.** Deriving Security Requirements from Business Process Models. Tartu, 2014, 171 p.
94. **Liina Kamm.** Privacy-preserving statistical analysis using secure multi-party computation. Tartu, 2015, 201 p.
100. **Abel Armas Cervantes.** Diagnosing Behavioral Differences between Business Process Models. Tartu, 2015, 193 p.
101. **Fredrik Milani.** On Sub-Processes, Process Variation and their Interplay: An Integrated Divide-and-Conquer Method for Modeling Business Processes with Variation. Tartu, 2015, 164 p.
102. **Huber Raul Flores Macario.** Service-Oriented and Evidence-aware Mobile Cloud Computing. Tartu, 2015, 163 p.
103. **Tauno Metsalu.** Statistical analysis of multivariate data in bioinformatics. Tartu, 2016, 197 p.
104. **Riivo Talviste.** Applying Secure Multi-party Computation in Practice. Tartu, 2016, 144 p.
108. **Siim Orasmaa.** Explorations of the Problem of Broad-coverage and General Domain Event Analysis: The Estonian Experience. Tartu, 2016, 186 p.
109. **Prastudy Mungkas Fauzi.** Efficient Non-interactive Zero-knowledge Protocols in the CRS Model. Tartu, 2017, 193 p.
110. **Pelle Jakovits.** Adapting Scientific Computing Algorithms to Distributed Computing Frameworks. Tartu, 2017, 168 p.
111. **Anna Leontjeva.** Using Generative Models to Combine Static and Sequential Features for Classification. Tartu, 2017, 167 p.
112. **Mozhgan Pourmoradnasseri.** Some Problems Related to Extensions of Polytopes. Tartu, 2017, 168 p.

113. **Jaak Randmets.** Programming Languages for Secure Multi-party Computation Application Development. Tartu, 2017, 172 p.
114. **Alisa Pankova.** Efficient Multiparty Computation Secure against Covert and Active Adversaries. Tartu, 2017, 316 p.
116. **Toomas Saarsen.** On the Structure and Use of Process Models and Their Interplay. Tartu, 2017, 123 p.
121. **Kristjan Korjus.** Analyzing EEG Data and Improving Data Partitioning for Machine Learning Algorithms. Tartu, 2017, 106 p.
122. **Eno Tõnisson.** Differences between Expected Answers and the Answers Offered by Computer Algebra Systems to School Mathematics Equations. Tartu, 2017, 195 p.

DISSERTATIONES INFORMATICAE UNIVERSITATIS TARTUENSIS

1. **Abdullah Makkeh**. Applications of Optimization in Some Complex Systems. Tartu 2018, 179 p.
2. **Riivo Kikas**. Analysis of Issue and Dependency Management in Open-Source Software Projects. Tartu 2018, 115 p.
3. **Ehsan Ebrahimi**. Post-Quantum Security in the Presence of Superposition Queries. Tartu 2018, 200 p.
4. **Ilya Verenich**. Explainable Predictive Monitoring of Temporal Measures of Business Processes. Tartu 2019, 151 p.
5. **Yauhen Yakimenka**. Failure Structures of Message-Passing Algorithms in Erasure Decoding and Compressed Sensing. Tartu 2019, 134 p.
6. **Irene Teinemaa**. Predictive and Prescriptive Monitoring of Business Process Outcomes. Tartu 2019, 196 p.
7. **Mohan Liyanage.** A Framework for Mobile Web of Things. Tartu 2019, 131 p.
8. **Toomas Krips.** Improving performance of secure real-number operations. Tartu 2019, 146 p.
9. **Vijayachitra Modhukur.** Profiling of DNA methylation patterns as biomarkers of human disease. Tartu 2019, 134 p.
10. **Elena Sügis.** Integration Methods for Heterogeneous Biological Data. Tartu 2019, 250 p.
11. **Tõnis Tasa.** Bioinformatics Approaches in Personalised Pharmacotherapy. Tartu 2019, 150 p.
12. **Sulev Reisberg.** Developing Computational Solutions for Personalized Medicine. Tartu 2019, 126 p.
13. **Huishi Yin.** Using a Kano-like Model to Facilitate Open Innovation in Requirements Engineering. Tartu 2019, 129 p.
14. **Faiz Ali Shah.** Extracting Information from App Reviews to Facilitate Software Development Activities. Tartu 2020, 149 p.
15. **Adriano Augusto**. Accurate and Efficient Discovery of Process Models from Event Logs. Tartu 2020, 194 p.
16. **Karim Baghery.** Reducing Trust and Improving Security in zk-SNARKs and Commitments. Tartu 2020, 245 p.
17. **Behzad Abdolmaleki.** On Succinct Non-Interactive Zero-Knowledge Protocols Under Weaker Trust Assumptions. Tartu 2020, 209 p.
18. **Janno Siim.** Non-Interactive Shuffle Arguments. Tartu 2020, 154 p.
19. **Ilya Kuzovkin.** Understanding Information Processing in Human Brain by Interpreting Machine Learning Models. Tartu 2020, 149 p.
20. **Orlenys López Pintado.** Collaborative Business Process Execution on the Blockchain: The Caterpillar System. Tartu 2020, 170 p.
21. **Ardi Tampuu.** Neural Networks for Analyzing Biological Data. Tartu 2020, 152 p.

22. **Madis Vasser.** Testing a Computational Theory of Brain Functioning with Virtual Reality. Tartu 2020, 106 p.
23. **Ljubov Jaanuska.** Haar Wavelet Method for Vibration Analysis of Beams and Parameter Quantification. Tartu 2021, 192 p.
24. **Arnis Parsovs.** Estonian Electronic Identity Card and its Security Challenges. Tartu 2021, 214 p.
25. **Kaido Lepik.** Inferring causality between transcriptome and complex traits. Tartu 2021, 224 p.
26. **Tauno Palts.** A Model for Assessing Computational Thinking Skills. Tartu 2021, 134 p.
27. **Liis Kolberg.** Developing and applying bioinformatics tools for gene expression data interpretation. Tartu 2021, 195 p.
28. **Dmytro Fishman.** Developing a data analysis pipeline for automated protein profiling in immunology. Tartu 2021, 155 p.