IMPROVEMENTS IN COMMUNICATION COMPLEXITY USING QUANTUM ENTANGLEMENT

A Thesis

by

ANGAD MOHANDAS KAMAT

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2008

Major Subject: Computer Science

IMPROVEMENTS IN COMMUNICATION COMPLEXITY USING QUANTUM ENTANGLEMENT

A Thesis

by

ANGAD MOHANDAS KAMAT

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Andreas Klappenecker
Committee Members,	Jennifer L. Welch
	Alexander Sprintson
Head of Department,	Valerie E. Taylor

August 2008

Major Subject: Computer Science

ABSTRACT

Improvements in Communication Complexity
Using Quantum Entanglement. (August 2008)

Angad Mohandas Kamat, B. Tech., National Institute of Technology,
Warangal, India
Chair of Advisory Committee: Dr. Andreas Klappenecker

Quantum computing resources have been known to provide speed-ups in computational complexity in many algorithms. The impact of these resources in communication, however, has not attracted much attention. We investigate the impact of quantum entanglement on communication complexity. We provide a positive result, by presenting a class of multi-party communication problems wherein the presence of a suitable quantum entanglement lowers the classical communication complexity. We show that, in evaluating certains function whose parameters are distributed among various parties, the presence of prior entanglement can help in reducing the required communication. We also present an outline of realizing the required entanglement through optical photon quantum computing. We also suggest the possible impact of our results on network information flow problems, by showing an instance of a lower bound which can be broken by adding limited power to the communication model.

To Aie and Baba, for always being there

ACKNOWLEDGMENTS

This work would not have been realized without constant guidance and inspiration from Dr. Andreas Klappenecker. I would like to thank him for introducing me to the exciting world of quantum computing, and for supporting my research. I would also like to thank Dr. Jennifer Welch for her encouragement; her course on distributed systems gave me the best possible introduction to abstract thinking and proof techniques in graduate school. My gratitude also extends to Dr. Alexander Sprintson for providing food for thought through various problems in network information flow. I would also like to thank my colleague Pradeep Kiran Sarvepalli who has always been willing to respond to my innumerable questions and discussions. This research was supported by NSF CAREER Award CCF-0347310, NSF Grant CCF-0622201, and a TITF project.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

TABLE Page

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The notion of computability, as suggested by the strong Church-Turing thesis [16], has been shaken up ever since the advent of the quantum computing model. Having shown exponential gains in asymptotic execution times of some classical algorithms, quantum computing has provided a new outlook for the canonical notion of efficiency. Most importantly, quantum complexity theory has armed us with novel techniques for answering *the* question of computer science, $P \stackrel{?}{=} NP$.

From a complexity perspective, this thesis investigates the impact of quantum computing on communication problems. It is observed that the presence of quantum computing resources helps reduce communication required for solving certain problems. The improvements shown thus extend the positive impact of quantum computing to some scenarios of distributed computing.

This thesis is organized as follows. In the next chapter, we present a four-party communication problem which shows superior communication complexity when aided by entanglement. We then follow up with a generalization to m-party communication problems which show similar superiority. Then, we briefly discuss implementation of the required quantum entanglement using optical photon quantum computing. We conclude by discussing some implications of our results on network information flow and enlisting some future directions.

In this chapter, we first present brief introductions to quantum computing and communication complexity. We then provide some motivation for pursuit of the problem and summarize some of the related research. Finally, we include a section

The journal model is *IEEE Transactions on Information Theory.*

that enlists the notation used in the rest of the thesis.

A. Quantum computing and entanglement

Just as any classical two-level system represents a classical bit, a two-level quantum mechanical system can be used to represent a quantum bit (*qubit* for short). These two levels of a qubit, called the *basis states*, are represented as $|0\rangle$ and $|1\rangle$. However, unlike that of a classical bit, the description of a qubit includes probabilities associated with the two states. Specifically, a qubit $|a\rangle$ may be represented as a *superposition*

$$|a\rangle = a_0|0\rangle + a_1|1\rangle, \text{ such that } a_0, a_1 \in \mathbb{C}, \; |a_0|^2 + |a_1|^2 = 1.$$

The probability of $|a\rangle$ being in state $|i\rangle$ is $|a_i|^2$. When measurement of a qubit is performed, we observe either 0 or 1, depending upon their probability distribution. Mathematically, the basis states $|0\rangle$ and $|1\rangle$ are column vectors $\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$ and $\left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]$ respectively; the qubit $|a\rangle$ is $\left[\begin{smallmatrix}a_0\\a_1\end{smallmatrix}\right] \in \mathbb{C}^2$.

When we consider the joint state of two or more bits, it can be expressed merely by their concatenation. On the other hand, the joint state of multiple qubits is expressed as their *tensor product*. For example, the joint state of $|a\rangle$ and $|b\rangle = b_0|0\rangle + b_1|1\rangle$ is

$$|a\rangle \otimes |b\rangle = a_0b_0|00\rangle + a_0b_1|01\rangle + a_1b_0|10\rangle + a_1b_1|11\rangle.$$

This exponential growth in the number of basis states provides the parallelism for gains in time complexity.

Analogous to logic gates, quantum computing has quantum gates. All operations on qubits are linear, and can be represented as unitary[1] transforms. By definition,

[1] A unitary matrix is one whose inverse is its conjugate transpose.

a unitary transform preserves orthogonality of quantum states. An example is the Hadamard transform $H = \frac{1}{\sqrt{2}}\left[\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right]$, which we shall encounter frequently. Clearly, its action is

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle),\ H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

In some cases, the quantum state of multiple qubits cannot be expressed as the joint space of any individual qubits. For example, while we can express $\frac{1}{\sqrt{2}}(|00\rangle+|10\rangle)$ as $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle$, we cannot find such a separation for $\frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$. The latter is called an *entangled* state, owing to the fact that measurement of one qubit affects the outcome of measuring the other.

With respect to our results, quantum entanglement is the important resource that we use to exploit the correlation among measurements. We design an entangled state that follows a specific pattern of measurement outcomes suitable to solving our communication problem.

Readers interested in more details about quantum computation and information are encouraged to read [10], which provides a comprehensive overview.

B. Communication complexity

Communication complexity is the study of bounds on the total communication required to perform certain distributed tasks. The general communication problem is as follows – a system (a group of parties with communication channels connecting them) is required to evaluate a multi-parameter function, with the parameters being distributed among different parties. The *communication complexity* of the function in the given system is the minimum number of data units (typically, bits) that need to be sent over the communication channels in order for the function to be evaluated.

Communication complexity was formally introduced by Yao [17]. The system here comprised of two parties, Alice and Bob, possessing input strings x and y respectively. The goal is for Alice to evaluate

$$f(x,y),\ f:\{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\},$$

with the minimum possible communication. The minimum number of bits that Bob must send to help Alice evaluate f is the communication complexity of f. It is obvious that the communication complexity of any f is upper-bounded by n – Bob can simply send his entire input to Alice, who then evaluates f. There are functions which require way less communication than n. For instance, if

$$f(x,y) = (x+y) \bmod 2,$$

then Bob only needs to indicate to Alice whether x is odd or even, thus requiring a single bit of communication. However, there also exist functions, such as the equality operation

$$f(x,y) = (x \overset{?}{=} y),$$

where n bits of communication are necessary.

Despite being highly abstract, communication complexity has many applications. It is famously used in VLSI design, where communication complexity can be linked to circuit depth. The book [9] serves as an excellent survey of the developments in this field.

With respect to this thesis, we consider a four-party communication setting, and then extend it to multi-party communication problems. The goal is for one party to evaluate a multiple-argument function. Communication complexity now accounts for *any* communication that that takes place between any pair of parties.

C. Motivation and background

Ever since Peter Shor first introduced the polynomial-time factoring algorithm [15], most research in quantum algorithms has aggressively sought instances to prove tremendous asymptotic gains in running times. While no result of universality has been obtained in this direction, quantum computing resources have been able to provide speed-ups in some specific problems. Most of these gains have been confined to algorithm development; very few other domains have been able to show similar results. Our work explores such a gain in the domain of communication complexity. In solving certain problems that require communication among various parties, we observe that the availability of quantum entanglement reduces the communication necessary to solve the problem.

Quantum entanglement has been a very useful resource offered by quantum computing. The ability to exhibit the so-called "non-local" effects has given a unique power over classical computing, especially in a distributed setting. A canonical example of this power is shown by quantum teleportation [2], which makes use of the famous Bell state to transfer a qubit using only classical communication.

Quantum entanglement, however, cannot be used as a medium of communication. Whenever prior entanglement is shared among parties, there is no way that two parties can exchange any information by merely operating upon the entangled qubits [6].

The utility of quantum entanglement in reducing communication complexity was first shown in [5]. It presents a three-party communication protocol wherein prior shared entanglement reduces the classical communication complexity of a certain function by one bit. Specifically, Alice, Bob and Carol each have n-bit input strings x, y and z, which are subject to the constraint

$$\forall j,\ x_j \oplus y_j \oplus z_j = 1.$$

The goal is for Alice to evaluate

$$f(x, y, z) = \bigoplus_{j=1}^{n} x_j y_j z_j$$

using minimum possible communication with Bob and Carol. The authors show that using entanglements of the form

$$\frac{1}{2}(|001\rangle + |010\rangle + |100\rangle - |111\rangle),$$

two bits of communication are sufficient for Alice to compute f. On the other hand, it is shown that three bits of communication are necessary (and sufficient) in the absence of quantum entanglement.

Similar results were shown in [3], which also explored a two-party probabilistic communication protocol. It was shown that prior entanglement achieves a certain probability of success with two bits of communication – something that a classical shared random string cannot. Such a result also shows that entanglement is more powerful than a shared random variable.

This thesis gives an extension to the results in [5], by presenting a four-party communication protocol that computes a certain function using only three bits of communication when aided by a specific entanglement. It is also shown that without quantum entanglement, even four bits are insufficient to compute the same function. The problem is the generalized to a class of m-party communication problems which show similar gains in communication complexity – the presence of a suitable entangled state almost halves the communication complexity. Finally, we also discuss implementation of the quantum circuits required for the entanglement and the realization of the circuits using optical photon quantum computing.

D. Notation

A finite field of of size 2 is denoted by $\mathbb{F}_2$, while $\mathbb{F}_2^m$ denotes the set of m-bit strings. For any bit string x, x_i denotes its i-th bit. For any $q \in \mathbb{F}_2^m$, $|q| = \sum_{i=1}^m q_i$ denotes the Hamming weight of q, and $|q|_i = |q| \bmod i$ denotes the Hamming weight modulo-i. When the set $\mathbb{F}_2^m$ is clear from context, $\hat{0}$ and $\hat{1}$ represent the bit strings with all positions 0 and 1, respectively.

The Knuth-Iverson bracket is denoted by $[stmt]$; it evaluates to 1 if the statement *stmt* is true, and to 0 otherwise. For any boolean variable y, $N_0(y) = y$ and $N_1(y) = \overline{y}$, respectively denoting the logical identity and negation operations.

Let I be a subset of $\{1, \ldots, m\}$ of cardinality ℓ. We define a function $\pi_I \colon \mathbb{F}_2^m \times \mathbb{F}_2^\ell \to \mathbb{F}_2^m$ by

$$\pi_I((a_1, \ldots, a_m), (b_k)_{k \in I}) = (a_i[i \notin I] + b_i[i \in I])_{i=1,\ldots,m},$$

that is, π_I replaces the elements in the vector $(a_1, \ldots, a_m)$ at positions in I by elements of the vector $(b_k)_{k \in I}$. $\chi(I)$ denotes the characteristic vector of the index set I, that is, $\chi(I) = ([i \in I])_{i=1,\ldots,m} \in \mathbb{F}_2^m$.

If U denotes a unitary transform[2] on a single qubit and c a vector in $\mathbb{F}_2^m$, then we denote by U^c the matrix $U^{c_1} \otimes \cdots \otimes U^{c_m}$.

[2] This is nothing but a 2 × 2 unitary matrix.

CHAPTER II

FOUR-PARTY COMMUNICATION PROBLEM

In this chapter, we introduce a four-party communication problem for the computation of a four-argument boolean function. The goal is for one of the parties to evaluate the function provided that the inputs satisfy a certain constraint. We perform an analysis of the communication complexity, both with and without the availability of prior entanglement. While the best protocol without entanglement requires 5 bits of communication, an entanglement-assisted protocol can achieve the same goal in 3.

A. Problem description and communication model

The communication setting that we consider is comprised of four parties, connected by communication channels in a star network. Alice, Bob, Carol and Dan have n-bit input strings x^1, x^2, x^3 and x^4 respectively, and Alice is the hub of the network. We shall refer to the parties as A, B, C and D (they are also enumerated 1 through 4 in this order).

The inputs of the parties are subject to the condition

$$\forall j,\ x^1_j \oplus x^2_j \oplus x^3_j \oplus x^4_j = 0 \tag{2.1}$$

where $\oplus$ represents bitwise modulo-2 addition. The goal is for Alice to evaluate

$$f(x^1, x^2, x^3, x^4) = \bigoplus_{j=1}^{n} (x^1_j \cdot x^2_j \cdot x^3_j \cdot x^4_j \vee \overline{x^1_j} \cdot \overline{x^2_j} \cdot \overline{x^3_j} \cdot \overline{x^4_j}). \tag{2.2}$$

Owing to the precondition on the inputs, Alice is required to correctly evaluate f only when the condition is satisfied.

For the sake of simplicity, we assume a star network of communication. We later argue that the same complexity bounds hold in a complete graph network as well.

B. Entanglement-assisted protocol

1. Protocol description

For the sake of evaluating f, we propose that the four parties share a prior quantum entanglement of four qubits, with a total of n such entanglements for every corresponding input index. Let q_j^i be the j-th qubit of a party i, where $j \in \{1, \ldots, n\}$. For each j, the quadruplet of qubits is in the entangled state

$$\begin{aligned} |q_j^1 q_j^2 q_j^3 q_j^4\rangle = \tfrac{1}{2\sqrt{2}} \big(&|0001\rangle + |0010\rangle + |0100\rangle + |1000\rangle \\ &-|1110\rangle - |1101\rangle - |1011\rangle - |0111\rangle\big). \end{aligned} \tag{2.3}$$

The entire state of all qubits is $|\psi\rangle = \bigotimes_{j=1}^{n} |q_j^1 q_j^2 q_j^3 q_j^4\rangle$.

Armed with this prior entanglement, each party i runs the simple protocol in Fig. 1, as a part of evaluating f. We shall call this *Protocol Q*.

```
1.     for each $j \in \{1, \ldots, n\}$, do
1.1        if $x_j^i = 0$, apply $H$ to $q_j^i$
1.2        measure $q_j^i$ giving the bit $s_j^i$
2.     $s^i \leftarrow \bigoplus_{j=1}^{n} s_j^i$
```

Fig. 1. Protocol Q.

In protocol Q, H is the usual Hadamard transform. The measurements are performed in the standard basis $\{|0\rangle, |1\rangle\}$. After the protocol is executed, B, C, D send their respective bits s^2, s^3, s^4 to A, who then computes $\oplus_{i=1}^{4} s^i$.

2. Correctness of the protocol

We now prove the correctness of the protocol, by beginning the the following crucial lemma.

Lemma 1. *For all $j \in \{1, \ldots, n\}$, we have*

$$s_j^1 \oplus s_j^2 \oplus s_j^3 \oplus s_j^4 = x_j^1 \cdot x_j^2 \cdot x_j^3 \cdot x_j^4 \vee \overline{x_j^1} \cdot \overline{x_j^2} \cdot \overline{x_j^3} \cdot \overline{x_j^4}.$$

Proof. By the condition (2.1), we observe that

$$x_j^1 x_j^2 x_j^3 x_j^4 \in \{0000, 0011, 0101, 0110, 1001, 1010, 1100, 1111\}. \tag{2.4}$$

In other words, corresponding to a $j \in \{1, \ldots, n\}$, a quadruplet of input bits only contains an even number of zeros. Accordingly, (2.3) is acted upon by 0, 2 or 4 H-gates. We now perform a case-by-case analysis to complete the proof.

- *Case 0.* Since no H-gates are applied, measurement of state (2.3) leads to

$$s_j^1 \oplus s_j^2 \oplus s_j^3 \oplus s_j^4 = 1 = x_j^1 \cdot x_j^2 \cdot x_j^3 \cdot x_j^4 \vee \overline{x_j^1} \cdot \overline{x_j^2} \cdot \overline{x_j^3} \cdot \overline{x_j^4}.$$

- *Case 4.* This happens when we have $x_j^1 x_j^2 x_j^3 x_j^4 = 0000$. In this case, (2.3) is acted upon trivially:

$$(H \otimes H \otimes H \otimes H)|q_j^1 q_j^2 q_j^3 q_j^4\rangle = |q_j^1 q_j^2 q_j^3 q_j^4\rangle.$$

  Measurement outcome is the same as in *Case 0.*

- *Case 2.* All these cases have a common property:

$$x_j^1 \cdot x_j^2 \cdot x_j^3 \cdot x_j^4 \vee \overline{x_j^1} \cdot \overline{x_j^2} \cdot \overline{x_j^3} \cdot \overline{x_j^4} = 0. \tag{2.5}$$

  Consider $x_j^1 x_j^2 x_j^3 x_j^4 = 0011$. Here, the H-gate is applied by A and B only. Thus, we get

$$H \otimes H \otimes I \otimes I|q_j^1 q_j^2 q_j^3 q_j^4\rangle = \tfrac{1}{2\sqrt{2}}\,(|0000\rangle - |0011\rangle + |0101\rangle + |0110\rangle$$
$$+|1001\rangle + |1010\rangle - |1100\rangle + |1111\rangle).$$

After this transformation, any measurement outcome yields $s_j^1 \oplus s_j^2 \oplus s_j^3 \oplus s_j^4 = 0$. Thus by (2.5), the lemma is satisfied. The remaining actions can easily be proved by symmetry to this case.

□

The success of the protocol is now immediate from the following theorem.

Theorem II.1. *For s^i and x^i defined above with reference to protocol Q, we have*

$$s^1 \oplus s^2 \oplus s^3 \oplus s^4 = f(x^1, x^2, x^3, x^4).$$

Proof.

$$\begin{aligned} s^1 \oplus s^2 \oplus s^3 \oplus s^4 &= (\bigoplus_{i=1}^{n} s_j^1) \oplus (\bigoplus_{i=1}^{n} s_j^2) \oplus (\bigoplus_{i=1}^{n} s_j^3) \oplus (\bigoplus_{i=1}^{n} s_j^4) \\ &= \bigoplus_{i=1}^{n} \left(s_j^1 \oplus s_j^2 \oplus s_j^3 \oplus s_j^4\right) \\ &= \bigoplus_{i=1}^{n} \left(x_j^1 \cdot x_j^2 \cdot x_j^3 \cdot x_j^4 \vee \overline{x_j^1} \cdot \overline{x_j^2} \cdot \overline{x_j^3} \cdot \overline{x_j^4}\right), \text{ from Lemma 1} \\ &= f(x^1, x^2, x^3, x^4). \end{aligned}$$

□

It may be noted that entanglement is an extra resource; yet, it does not provide any information gain with respect to the inputs x^i. The improvement in complexity arises from intelligent use of the input preconditions. Thus, as apparent from Protocol

Q and Theorem II.1, 3 bits of communication are sufficient, giving us an upper bound on entanglement-assisted communication complexity.

C. Purely classical protocol

We now prove that without prior entanglement, 5 bits of communication are necessary and sufficient for any protocol that evaluates (2.2) subject to (2.1). We complete this proof in two steps, by showing that

- there exists a purely classical protocol using 5 bits of communication (without any quantum entanglement resource), and
- no 4-bit classical protocol exists for solving this problem.

1. Classical protocol for upper bound

We now express f in terms of another function g, to help us design a communication protocol. Just like f, g is a function of the type

$$g : \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^n \mapsto \{0,1\}.$$

By expressing f in terms of g, we use the protocol for evaluating g to, in turn, evaluate f.

Let u^i be the number of zeros in the x^i. Define g as

$$g(x^1, x^2, x^3, x^4) = \frac{\left(\sum_i u^i\right) \bmod 4}{2}. \tag{2.6}$$

We must also ensure that input conditions are properly translated into this definition. Owing to (2.1), we get the following condition on u^i's.

$$\sum_{i=1}^{4} u^i \equiv 0 \pmod 2. \tag{2.7}$$

Notice that due to input condition, the numerator of g is always even; and being a modulo-4 number, it can be either 0 or 2. This rules out any inconsistencies (such as fractions) in the division.

For convenience, we represent these functions as f_n and g_n, where n is the length of their inputs. The following theorem shows the relation between f and g.

Theorem II.2. *For any $n \geq 1$ and for u^i defined above,*

$$\left(\sum_{i=1}^{4} u^i \equiv 0 \pmod 2 \right) \Rightarrow \left(f_n = g_n \oplus (n \bmod 2) \right).$$

Proof. We define the predicate

$$P(n) \overset{def}{:=} \left(\sum_{i=1}^{4} u^i \equiv 0 \pmod 2 \right) \Rightarrow \left(f_n = g_n \oplus (n \bmod 2) \right).$$

The truth of $P(n)$ is shown by induction on n. $P(1)$ can easily be proved from (2.4). For $n = 1$,

$$f_1 = x^1 \cdot x^2 \cdot x^3 \cdot x^4 \vee \overline{x^1} \cdot \overline{x^2} \cdot \overline{x^3} \cdot \overline{x^4}.$$

Also, (2.4) tells us that the total number of zeros can only be even. With precondition satisfied, f_1 evaluates to 1 if and only if the inputs are either all 0 or all 1. In these cases, g_1 evaluates as $0 \oplus 1 = 1$. In all other cases, $g_1 = 1 \oplus 1 = 0$, which is identical to the behavior of f_1.

By hypothesis, assume the truth of $P(k)$. Let

$$T_j = x_j^1 \cdot x_j^2 \cdot x_j^3 \cdot x_j^4 \vee \overline{x_j^1} \cdot \overline{x_j^2} \cdot \overline{x_j^3} \cdot \overline{x_j^4}.$$

We have

$$\begin{aligned} f_{k+1} &= f_k \oplus T_{k+1} \\ &= g_k \oplus \ (k \bmod 2) \oplus T_{k+1}, \text{ by hypothesis} \\ &= g_k \oplus \ (k+1 \bmod 2) \oplus 1 \oplus T_{k+1}. \end{aligned}$$

Owing to (2.4), we can only have the following cases.

- *T_{k+1} has two bits as zero.* Here, $T_{k+1} = 0$, $g_{k+1} = g_k \oplus 1$.
- *T_{k+1} has none or all four bits as zero.* Here, $T_{k+1} = 1$, $g_{k+1} = g_k$.

Both cases confirm the truth of $P(k+1)$. □

The 5-bit protocol now follows suite. Each of B, C and D finds its respective u^i. Since the evaluation of g involves the computation of sums modulo-4, the parties only need to consider the two least significant bits of u^i's. Moreover, due to (2.7), it is known that the sum of u^i's is even. The protocol now works as follows.

> Any two parties send the two least significant bits of u^i to A, while the third party sends the second-least significant bit of u^i. Using these bits, A can determine g, from which f can be evaluated by Theorem II.2.

2. Proof of lower bound

We make use of Theorem II.2 in our proof of lower bound. From the theorem, we can conclude that f_n and g_n have the same communication complexity[1]. We prove a lower bound the communication complexity of g, which also holds for f.

[1] Communication complexity is a property of the function. Since the communication model does not change between computing f and g, their communication complexities must be the same.

Our communication problem is now reduced to this: the four parties A, B, C and D each have a positive integer u^i as their input. We need to calculate g. We have a condition (2.7), which defines the valid set of inputs. Moreover, since we are interested in calculating modulo-4 sum, we can discard all but the two least significant bits in the binary representation of u^i. Hence, from now on, we assume that $u^i \in \{0, 1, 2, 3\}$.

By way of contradiction, suppose there exists a 4-bit classical protocol to evaluate g. For optimality, each of the parties B, C, D must send one bit to A, and any one must send another bit; A can then evaluate the function. Each party *must* send something, otherwise no information about its input can be conveyed to A. Without loss of generality, B sends two bits, and C, D send one bit each. Now, our problem is essentially that of *three-party* communication, since A knows the input of B.

The bits sent by C and D can be assumed to be functions of their inputs, of the following type.

$$\phi : \{0, 1, 2, 3\} \rightarrow \{0, 1\}.$$

Any ϕ partitions the input set into two, $S_0 = \phi^{-1}(0)$ and $S_1 = \phi^{-1}(1)$. Without loss of generality, let $0 \in S_0$. Clearly, there exists a unique S_0 for every ϕ, since ϕ can be deduced from S_0. A protocol is nothing but a combination of two such mappings ϕ, one each for C and D. We denote a *partitioning strategy* based on the above premise by (S^C, S^D), depending only on C and D.

We perform an exhaustive analysis to show that every combination of partitions by C and D fails to convey enough information for A to evaluate g, subject to (2.7). The basic idea is as follows.

> When A knows the partitions that each input belongs to, she determines all the input combinations of the four parties. She then determines the values of g for each valid input combination (validity of inputs as de-

fined by (2.7)). If all the possible function values are identical, then this partitioning strategy yields a successful protocol; otherwise, it fails.

The following theorem covers the proof of impossibility.

Theorem II.3. *There exists no partitioning strategy* (S^C, S^D) *to compute* g *of (2.6) subject to condition imposed by (2.7).*

Proof. Our proof strategy is as follows. First, we list all possible partitions of the input set $\{0, 1, 2, 3\}$. We then exaustively show that for every combination of these partitions, A does not have sufficient information to conclude about the value of g: in every combination, there exists a pair of "conflicting" inputs which give different function values.

There are 7 ways to partition the input set into two. Since C and D each has its own parition, we have to examine $7 \times 7 = 49$ combinations. By grouping, we present 5 cases.

We first list the 7 possible paritions, generated using the algorithm in [7]. The partitions are enumerated in Table I.

Table I. Partitions of the four-element set into two.

	S_0	S_1
p_1	$\{0,2,3\}$	$\{1\}$
p_2	$\{0,2\}$	$\{1,3\}$
p_3	$\{0,3\}$	$\{1,2\}$
p_4	$\{0\}$	$\{1,2,3\}$
p_5	$\{0,1,3\}$	$\{2\}$
p_6	$\{0,1\}$	$\{2,3\}$
p_7	$\{0,1,2\}$	$\{3\}$

With this enumeration, we show the ambiguous inputs for every combination of partitions. In Table II, the columns S^P show the p_i's used by P; $*$ indicates that any p_i may be used. The columns $g = 0$ and $g = 1$ show the inputs which are indistinguishable to A; yet they give different function values.

Table II. No partitioning strategy leads to a protocol.

#	S^C	S^D	$g = 0$	$g = 1$	Cases
1	$*$	p_1, p_2, p_7	$(0, 0, 0, 0)$	$(0, 0, 0, 2)$	21
2	$*$	p_4, p_5	$(1, 0, 0, 3)$	$(1, 0, 0, 1)$	14
3	p_1, p_2	p_3, p_6	$(0, 0, 0, 0)$	$(0, 0, 2, 0)$	4
4.1	p_5, p_6, p_7	p_3	$(1, 0, 0, 3)$	$(1, 0, 1, 0)$	3
4.2		p_6	$(0, 0, 0, 0)$	$(0, 0, 1, 1)$	3
5.1	p_3, p_4	p_3	$(0, 0, 1, 3)$	$(0, 0, 2, 0)$	2
5.2		p_6	$(1, 0, 2, 1)$	$(1, 0, 1, 0)$	2

For the sake of completeness, we explain how our reasoning works, using one of the cases listed in the table. Consider case 1, which says that C can follow any partitioning strategy, while D may follow any one of p_1, p_2 or p_7. In such a scenario, consider the input tuple (u^1, u^2, u^3, u^4) of g to be $(0, 0, 0, 0)$. The function value for this input is 0. Now consider the input tuple $(0, 0, 0, 2)$, whose function value is 1. Both are valid sets of inputs. However, with any combination of partitions we mentioned earlier, these inputs are indistinguishable. Thus, in light of such a protocol, A cannot successfully arrive at the correct function value because her knowledge about the inputs is insufficient.

In summary, for every strategy (S^C, S^D), there exist ambiguous values for g, making it impossible for A to compute it. □

There exists no classical protocol in a star network to evaluate f using 4 bits of communication. This gives us the tight bound of 5 bits on communication complexity.

3. Generalization of bounds to all network topologies

We now consider the case of a complete graph, where any party can communicate with any other. The problem at hand is that there are three parties (A and B can be considered as one), and two bits of communication should be able to help A evaluate g. The graph that we used to prove the lower bound was a star network; and communication occurred as in Fig. 2. The unlabeled arrows indicate single bit of communication. We now enlist the other possible communication scenarios with total two bits of communication among AB, C and D. They are captured in Fig. 3.

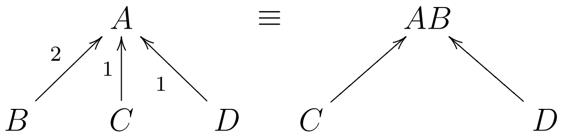

Fig. 2. Communication graph for 4-bit lower bound.

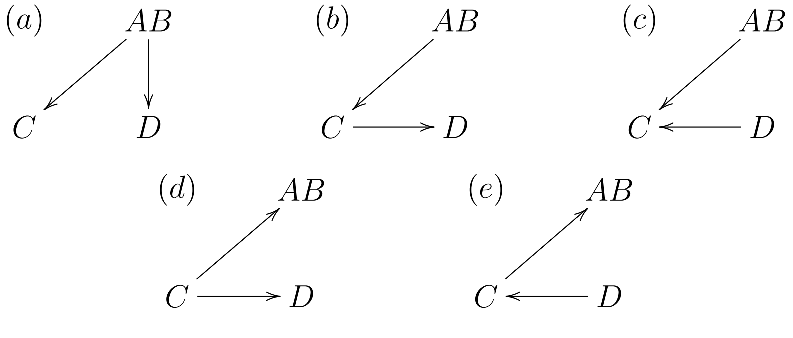

Fig. 3. Possible communication graphs for 4-bit scenario.

From the figure, cases (a), (b) and (c) do not lead to any protocol, since A receives no information about the inputs of C and D. These cases can be discarded right away. We are left with (d) and (e), where A receives one bit of communication.

In (d), C sends out two bits, and we can assume two mappings of type ϕ for these two bits. Irrespective of the chosen mapping, A has no way of knowing anything about the input of D, since it does not send any bits. Therefore, this communication is futile as well.

We turn to (e), which is special because there exists a path from both C and D to A. For a protocol to work, we would now require C to do some local computations before sending a bit to A. We can assume that D sends the bit according to the mapping ϕ_D based on its input. This arms C with more information than just her own input, and the bit sent from C to A comes from a mapping of the type

$$\phi_C : \{0, 1, 2, 3\} \times \{0, 1\} \to \{0, 1\}.$$

C now knows the partition to which D's input belongs, and she can include that information in the bit it sends to A. Such a strategy can clearly be no better than D sending a bit directly to A.

From the above arguments, we conclude that irrespective of the nature of the communication graph, the communication complexity of f is 5 bits.

CHAPTER III

GENERAL MULTI-PARTY COMMUNICATION PROBLEMS

In this chapter, we construct a family of multi-party communication problems, wherein the presence of a suitable quantum entanglement gives a communication complexity lower than that in the purely classical case. We first present a problem description in terms of the number of parties m and the input length n. We then explain the action of Hadamard gates on the entanglement, as relevant to the functioning of the entanglement-assisted protocol. Thereafter, we prove the upper and lower bounds on communication complexity in the absence of quantum entanglement.

The parties are enumerated 1 to m, with x^i denoting the input string of party i. The inputs are of length n. Each party also has n qubits, and the j-th qubit of i-th party is denoted by q^i_j. For $q \in \mathbb{F}_2^m$, we define the function $u : \mathbb{F}_2^m \to \{0, 1, -1\}$ as

$$u(q) = |q|_2(-1)^{[|q|_4=3]}.$$

A. Problem formulation

The following clauses define the m-party communication problem, for $m \geq 4$.

- *Input Condition.* The inputs satisfy the following constraint.

$$\forall j, \bigoplus_{i=1}^{m} x^i_j = m \bmod 2. \tag{3.1}$$

- *Entanglement.* The entanglment of all qubits of the same index is

$$|\varphi\rangle = \frac{1}{(\sqrt{2})^{m-1}} \sum_{q \in \mathbb{F}_2^m} u(q)|q\rangle \tag{3.2}$$

  where $|q^1_k q^2_k \ldots q^m_k\rangle$ is represented by $|q\rangle$.

- *Evaluation Function.* Party 1 is required to evaluate

$$f(x^1, \ldots, x^m) = \bigoplus_{j=1}^{n} T_j, \text{ where } T_j = \bigvee_{\substack{I \subseteq \{1,\ldots,m\} \\ |I| \equiv 0 \pmod 4}} \left(\bigwedge_{i=1}^{m} N_{[i \in I]}(x_j^i) \right). \tag{3.3}$$

We give an example for constructing f. For $m = 5$, we know that there are 6 sets I such that $|I| \equiv 0 \pmod 4$. Each such set gives one conjunction term, and T_j is obtained as

$$\begin{aligned} x_j^1 \cdot x_j^2 \cdot x_j^3 \cdot x_j^4 \cdot x_j^5 \quad & \vee \quad \overline{x_j^1} \cdot \overline{x_j^2} \cdot \overline{x_j^3} \cdot \overline{x_j^4} \cdot x_j^5 \\ & \vee \quad \overline{x_j^1} \cdot \overline{x_j^2} \cdot \overline{x_j^3} \cdot x_j^4 \cdot \overline{x_j^5} \\ & \vee \quad \overline{x_j^1} \cdot \overline{x_j^2} \cdot x_j^3 \cdot \overline{x_j^4} \cdot \overline{x_j^5} \\ & \vee \quad \overline{x_j^1} \cdot x_j^2 \cdot \overline{x_j^3} \cdot \overline{x_j^4} \cdot \overline{x_j^5} \\ & \vee \quad x_j^1 \cdot \overline{x_j^2} \cdot \overline{x_j^3} \cdot \overline{x_j^4} \cdot \overline{x_j^5}. \end{aligned}$$

As observed, among the m variables, each conjunction contains divisible-by-4 number of variables complemented.

For simplicity of argument, we assume a star network for communication, with party 1 as the hub of the network. We later prove that the bound holds for all communication topologies.

B. Entanglement-assisted protocol

With entanglement assistance, the function f is evaluated as follows. Every party first runs Protocol Q, and all but party 1 send their s^i values to party 1. Finally, party 1 computes the value of f as $\oplus_{i=1}^{m} s^i$. Thus, an m-party protocol with prior entanglement has a classical communication complexity of $m - 1$ bits.

The remainder of the section proves the correctness of this protocol.

1. More about the entanglement

We now unravel some facts about the entanglement (3.2). We formalize the possible actions of Hadamard gates onto $|\varphi\rangle$, to see its impact on the functioning of the protocol.

The following lemma shows the action of Hadamard gates on a basis state of qubits.

Lemma 2. *Let $q_k^1 \cdots q_k^m = q \in \mathbb{F}_2^m$. Let $I \subseteq \{1, \ldots, m\}$, $|I| = \ell$. If $c = \chi(I)$, then*

$$H^c|q_k^1 \cdots q_k^m\rangle = \frac{1}{(\sqrt{2})^\ell} \sum_{b \in \mathbb{F}_2^\ell} (-1)^{\sum_{i \in I} q_k^i b_i} |\pi_I(q, b)\rangle.$$

Now, we show some properties of a function $v : \mathbb{F}_2^{m-\ell} \times \mathbb{F}_2^\ell \to \mathbb{Z}$, defined as

$$v(p, b) = \sum_{r \in \mathbb{F}_2^\ell} u(p||r)(-1)^{r \cdot b} \tag{3.4}$$

where $||$ represents the concatenation operator. This function helps us determine some actions of Hadamard gates on the entanglement (3.2).

Lemma 3. *For u, v defined earlier, and for $0 \leq \ell \leq m$, we have*

$$\ell \equiv 0 \pmod 4 \Rightarrow |(p||b)|_2 = 0 \Leftrightarrow v(p, b) = 0,$$

$$\ell \equiv 2 \pmod 4 \Rightarrow |(p||b)|_2 = 0 \Leftrightarrow v(p, b) \neq 0.$$

Proof. We can rewrite v as follows.

$$\begin{aligned} v(p, b) &= \sum_{r \in \mathbb{F}_2^\ell} u(p||r)(-1)^{r \cdot b} \\ &= \sum_{\substack{r \in \mathbb{F}_2^\ell \\ |(p||r)|_4 = 1}} (-1)^{r \cdot b} - \sum_{\substack{r \in \mathbb{F}_2^\ell \\ |(p||r)|_4 = 3}} (-1)^{r \cdot b}. \end{aligned}$$

The above sums can be evaluated if we determine the sizes of orthogonal and non-orthogonal subspaces of b. We do this in Appendix A, for two separate cases. When $b \notin \{\hat{0}, \hat{1}\}$, we consult Lemma 8; otherwise, we use Lemma 9.

As per the notation in Appendix A, let B^0 and B^1 represent the orthogonal and non-orthogonal sub-spaces of F_2^ℓ with respect to b. Also, $B_{i,j}^k$ represents the proper subset with the extra condition,

$$B_{i,j}^k = \{r : r \in B^k, |r|_j = i\}.$$

We can now rewrite the above sums as follows.

$$\begin{aligned} v(p,b) &= \sum_{\substack{r \in \mathbb{F}_2^\ell \\ |(p||r)|_4 = 1}} (-1)^{r \cdot b} - \sum_{\substack{r \in \mathbb{F}_2^\ell \\ |(p||r)|_4 = 3}} (-1)^{r \cdot b} \\ &= \pm\big((|B_{w,4}^0| - |B_{w,4}^1|) - (|B_{w+2,4}^0| - |B_{w+2,4}^1|)\big) \end{aligned}$$

where w is determined by the Hamming weight of p. We now proceed with the proof by a case-by-case analysis.

The following results are evident when $b \notin \{\hat{0}, \hat{1}\}$.

1. Case $\ell \equiv 0 \pmod 4$. Here, we have from Lemma 8

$$|B_{w,4}^0| - |B_{w,4}^1| = 2^{\frac{\ell}{2}-1}(-1)^{\frac{\ell}{4}} \cos(\frac{w + |b|_4}{2}\pi).$$

By evaluating using this formula, we get the following results.

(a) $(|p|_2 = 0$ and $|b|_2 = 0)$ or $(|p|_2 = 1$ and $|b|_2 = 1) \Rightarrow (w + |b|_4) \equiv 1 \pmod 2$. This in turn means $u(p||b) = 0 \Rightarrow v(p,b) = 0$.

(b) $(|p|_2 = 1$ and $|b|_2 = 0)$ or $(|p|_2 = 0$ and $|b|_2 = 1) \Rightarrow (w + |b|_4) \equiv 0 \pmod 2$. By simplifying above equation, we get $v(p,b) = \pm 2^{\frac{\ell}{2}}(-1)^{\frac{\ell}{4}}$. Thus, $u(p||b) \neq 0 \Rightarrow v(p,b) \neq 0$.

Together, the two cases give $u(p||b) = 0 \Leftrightarrow v(p,b) = 0$.

2. Case $\ell \equiv 2 \pmod 4$. Here, we have from Lemma 8

$$|B^0_{w,4}| - |B^1_{w,4}| = 2^{\frac{\ell}{2}-1}(-1)^{\frac{\ell-2}{4}} \sin(\frac{w+|b|_4}{2}\pi).$$

By evaluating using this formula, we get the following results.

   (a) $(|p|_2 = 0$ and $|b|_2 = 0)$ or $(|p|_2 = 1$ and $|b|_2 = 1) \Rightarrow (w + |b|_4) \equiv 1 \pmod 2$. By simplifying above equation, we get $v(p, b) = \pm 2^{\frac{\ell}{2}}(-1)^{\frac{\ell-2}{4}}$. Thus, $u(p||b) = 0 \Rightarrow v(p, b) \neq 0$.

   (b) $(|p|_2 = 1$ and $|b|_2 = 0)$ or $(|p|_2 = 0$ and $|b|_2 = 1) \Rightarrow (w+|b|_4) \equiv 0 \pmod 2$. This in turn means $u(p||b) \neq 0 \Rightarrow v(p, b) = 0$.

   Together, the two cases give $u(p||b) = 0 \Leftrightarrow v(p, b) \neq 0$.

If $b \in \{\hat{0}, \hat{1}\}$, $|b|_2 = 0$ for $\ell \equiv 0 \pmod 2$. The following results are now evident from Lemma 9. We can perform a simplification of $v(p, b)$ just as above, but this time, the values can be obtained from (A.1).

1. Case $\ell \equiv 0 \pmod 4$.

   (a) $|p|_2 = 0 \Rightarrow v(p, b) = 0$. This in turn means $u(p||b) = 0 \Rightarrow v(p, b) = 0$.

   (b) $|p|_2 = 1 \Rightarrow v(p, b) = \pm(2)^{\frac{\ell}{2}}(-1)^{\frac{\ell}{4}}$. This in turn means $u(p||b) \neq 0 \Rightarrow v(p, b) \neq 0$.

   Together, the two cases give $u(p||b) = 0 \Leftrightarrow v(p, b) = 0$.

2. Case $\ell \equiv 2 \pmod 4$.

   (a) $|p|_2 = 0 \Rightarrow v(p, b) = \pm(2)^{\frac{\ell}{2}}(-1)^{\frac{\ell-2}{4}}$. This in turn means $u(p||b) = 0 \Rightarrow v(p, b) \neq 0$.

   (b) $|p|_2 = 1 \Rightarrow v(p, b) = 0$. This in turn means $u(p||b) \neq 0 \Rightarrow v(p, b) = 0$.

   Together, the two cases give $u(p||b) = 0 \Leftrightarrow v(p, b) \neq 0$.

□

We can now ascertain the action of Hadamard gates on $|\varphi\rangle$ defined in (3.2). This action is captured in the following theorem.

Theorem III.1. *Let $I \subseteq \{1, \ldots, m\}$ such that $|I| = \ell$, denote the ℓ positions where H-gates are applied on the $|\varphi\rangle$ defined in (3.2). If $c = \chi(I)$, then*

$$\ell \equiv 0 \pmod 4 \Rightarrow H^c|\varphi\rangle = \frac{1}{(\sqrt{2})^{m-1}} \sum_{q \in \mathbb{F}_2^m} (\pm|q|_2)|q\rangle,$$

$$\ell \equiv 2 \pmod 4 \Rightarrow H^c|\varphi\rangle = \frac{1}{(\sqrt{2})^{m-1}} \sum_{q \in \mathbb{F}_2^m} (\pm(1-|q|_2))|q\rangle.$$

Proof. For now, we will neglect the overall amplitude factor $\frac{1}{(\sqrt{2})^{m-1}}$. We shall take it into account at a later stage in the proof.

$$\begin{aligned} H^c|\varphi\rangle &= H^c \sum_{q \in \mathbb{F}_2^m} u(q)|q\rangle \\ &= \sum_{q \in \mathbb{F}_2^m} u(q) H^c |q\rangle \\ &= \sum_q \left(u(q) \sum_{b \in F_2^\ell} (-1)^{\sum_{i \in I} q_i b_i} |\pi_I(q,b)\rangle \right), \text{ by Lemma 2} \\ &= \sum_b \left(\sum_q u(q)(-1)^{\sum_{i \in I} q_i b_i} |\pi_I(q,b)\rangle \right). \end{aligned}$$

Let $\overline{I} = \{1, \ldots, m\} \setminus I$. Let $p \in F_2^{m-\ell}$ and $r \in F_2^\ell$, such that $q = \pi_I(q,r)$ and $q = \pi_{\overline{I}}(q,p)$. Note that $u(p||r) = u(\pi_{\overline{I}}(\pi_I(q,r),p))$. Now the above equation becomes

$$\begin{aligned} H^c|\varphi\rangle &= \sum_b \left(\sum_{p \in \mathbb{F}_2^{m-\ell}} \left(\sum_{r \in \mathbb{F}_2^\ell} u(p||r)(-1)^{r \cdot b} \right) |\pi_{\overline{I}}(\pi_I(q,b),p)\rangle \right) \\ &= \sum_b \left(\sum_{p \in \mathbb{F}_2^{m-\ell}} v(p,b) |\pi_{\overline{I}}(\pi_I(q,b),p)\rangle \right). \end{aligned}$$

Since the above sum ranges over $b \in \mathbb{F}_2^\ell$ and $p \in \mathbb{F}_2^{m-\ell}$, $q' = \pi_{\overline{I}}(\pi_I(q,b),p)$ clearly ranges over F_2^m. Also, $u(p,b) = u(q')$ and $v(p,b) = v(q')$. We get

$$H^c|\varphi\rangle = \sum_{q' \in \mathbb{F}_2^m} v(p,b)|q'\rangle.$$

We now account for the overall amplitude factor. After application of Hadamard gates, a factor of $\frac{1}{(\sqrt{2})^\ell}$ appears. The proof of Lemma 3 tells us that after the summation is performed, a factor of $\pm(\sqrt{2})^\ell$ appears. Thus, the two factors cancel out, and the proper sign remains with the corresponding basis state. The proof is now immediate from Lemma 3. □

2. Proof of correctness of m-party protocol

Having determined the behavior of the entanglement with Hadamard gates, we conclude the proof of correctness of the m-party protocol. We start with two simple observations that directly follow from (3.1).

Lemma 4. *Among the j-th input bits $\{x_j^1, x_j^2, \ldots, x_j^m\}$, the number of zero-bits is always even $\forall j$.*

This lemma directly implies the following, as evident from the operation of Protocol Q.

Lemma 5. *During the execution of Protocol Q, $|\varphi\rangle$ can only be acted upon by some $H^{\chi(I)}$ such that $|I| \equiv 0 \pmod 2$.*

The following is analogous to Lemma 1, and is the fundamental result behind the success of the protocol.

Lemma 6. *For all $j \in \{1, 2, \ldots, n\}$, we have*

$$\bigoplus_{i=1}^{m} s_j^i = T_j.$$

Proof. First off, we note that T_j evaluates to 1 if and only if exactly $l \equiv 0 \pmod 4$ of the x_j^i's in *some* conjunction are zero (see (3.3)), $1 \leq l \leq m$.

As protocol Q executes, H-gate is applied only when a party observes a 0. By Lemma 5, the number of H-gates (call it ℓ) applied on $|\varphi\rangle$ is always even.

The following two cases complete the proof, by Theorem III.1.

- $\ell \equiv 0 \pmod 4$. We know that $T_j = 1$. After tranformation, the amplitude of any basis state $|q'\rangle$ is non-zero if and only if $|q'|_2 \neq 0$. Thus, any measurement $|q'\rangle$ yields odd number of ones, giving $\bigoplus_{i=1}^m s_j^i = 1 = T_j$.
- $\ell \equiv 2 \pmod 4$. We know that $T_j = 0$. After tranformation, the amplitude of any basis state $|q'\rangle$ is non-zero if and only if $|q'|_2 = 0$. Thus, any measurement $|q'\rangle$ yields even number of ones, giving $\bigoplus_{i=1}^m s_j^i = 0 = T_j$.

□

Notice that Lemma 6 subsumes Lemma 1 of the four-party protocol. Thus on the lines of Theorem II.1, the correctness of the m-party protocol follows suite. We formally state it in the following theorem.

Theorem III.2. *For x^i and s^i defined in protocol Q, we have*

$$\bigoplus_{i=1}^{m} s^i = f(x^1, \ldots, x^m).$$

C. Bounds on purely classical protocol

As in the previous chapter, we now focus on proving lower and upper bounds on communication complexity in the absence of quantum entanglement. We observe that in this purely classical case, communication complexity is about double of that in the entanglement-assisted case. We follow the same technique as in the four-party case, by expressing f in terms of another function g and proving bounds for the latter.

1. Classical protocol for upper bound

Define u^i to be the number of zeros in x^i. Let

$$g(x^1, \ldots, x^m) = \frac{\left(\sum_{i=1}^m u^i\right) \bmod 4}{2}. \tag{3.5}$$

By Lemma 4, the input condition on x^i's translates to the following condition on u^i's.

$$\sum_{i=1}^{m} u^i \equiv 0 \pmod 2. \tag{3.6}$$

For convenience, we represent these functions as $f_{m,n}$ and $g_{m,n}$, where n is the length of the inputs and m is the number of parties. The following theorem shows the relation between f and g.

Theorem III.3. *For any $n \geq 1$, $m \geq 4$, and for u^i defined above,*

$$\left(\sum_{i=1}^{m} u^i \equiv 0 \pmod 2\right) \Rightarrow \left(f_{m,n} = g_{m,n} \oplus (n \bmod 2)\right).$$

Proof. We first define the predicate

$$P(n) \overset{def}{:=} \left(\sum_{i=1}^{m} u^i \equiv 0 \pmod 2\right) \Rightarrow \left(f_{m,n} = g_{m,n} \oplus (n \bmod 2)\right).$$

The proof is performed by induction on n.

For $P(1)$, $f_{m,1} = T_j$ from (3.3). Owing to Lemma 4, we only need to consider two cases such that $\sum_i u^i$ is (a) 0 mod 4, and (b) 2 mod 4. In (a), T_j is 1 and in (b), T_j is 0 (see Lemma 6), which in turn are the values of $f_{m,1}$. In the respective cases, $g_{m,1}$ takes the values 0 and 1. Thus, $g_{m,1} \oplus (1 \bmod 2)$ confirm the truth of $P(1)$.

By hypothesis, assume the truth of $P(j)$. We have

$$\begin{aligned} f_{m,j+1} &= f_{m,j} \oplus T_{j+1} \\ &= g_{m,j} \oplus \ (j \bmod 2) \oplus T_{j+1}, \text{ by hypothesis} \\ &= g_{m,j} \oplus \ (j+1 \bmod 2) \oplus 1 \oplus T_{j+1} \end{aligned}$$

Since T_{j+1} is has the same form as T_j, our analysis here is identical proving the basis. We have the same two cases (a) and (b) for T_{j+1}.

- In (a), $T_{j+1} = 1$. This means among the x^i_{j+1}'s, there are exactly $\ell \equiv 0 \pmod 4$ zero-bits. This keeps g unchanged, and $g_{m,j+1} = g_{m,j}$.
- In (b), $T_{j+1} = 0$. This means among the x^i_{j+1}'s, there are exactly $\ell \equiv 2 \pmod 4$ zero-bits. This causes g to flip, and $g_{m,j+1} = g_{m,j} \oplus 1$.

Both cases confirm the truth of $P(j+1)$. □

This gives an upper bound of $2(m-2)+1$ bits on communication complexity. Each party finds its respective u^P. All but two parties then send the two least significant bits of u^P to A, while remaining party sends the second-least significant bit of u^P. Using these bits, party 1 can determine g, from which f can be evaluated by Theorem III.3.

2. Proof of lower bound

Similar to the approach in Section C of the previous chapter, we prove a lower bound on $g_{m,n}$, which holds for $f_{m,n}$. We show that $2(m-2)$ bits of communication are insufficient to compute $f_{m,n}$.

Using Theorem III.3, we simplify the problem in the following manner: each party i has a positive integer u^i as input $(1 \leq i \leq m)$. Party 1 needs to calculate g defined in (3.5). Also, we have the condition on u^i, defined by (3.6). This further restricts the valid set of inputs. Moreover, as reasoned in Section C, we discard all but two least significant bits of u^i. Hence, $u^i \in \{0, 1, 2, 3\}$.

By contradiction, let there be a $2(m-2)$ bit classical protocol to evaluate (3.5). For optimality, each party must send at least a bit to party 1. Since $2(m-2)$ bits are communicated, by pigeon-hole principle, $m-3$ parties send their entire u^i's and the remaining two parties send one bit each.

We have now necessarily reduced our problem to three-party communication, just as in Section C of the previous chapter. From Theorem II.3, two bits are insufficient to solve this problem. Since $2(m-3)$ bits have already been communicated, $2(m-3)+2 = 2(m-2)$ bits are insufficient. Regarding $f_{m,n}$, we now have Theorem III.4.

Theorem III.4. *In a purely classical setting, the communication complexity of $f_{m,n}$ (subject to input conditions (3.1)) is $2(m-2)+1$ bits.*

Along with the upper bound, we get a tight bound of $2(m-2)+1$ bits on the communication complexity of $f_{m,n}$. This shows a clear separation between classical and entanglement-assisted communication complexities.

Similar to the analysis in the previous chapter, we can also argue that the complexity bound holds for all communication topologies among the m parties, because we have essentially reduced our communication problem to that of three parties.

CHAPTER IV

EXPERIMENTAL REALIZATION OF ENTANGLEMENT

We now turn to implementation details of the protocol, by specifically dealing with formulation of the entanglement. We first obtain an outline of the implementation using the quantum circuit model, and show how the circuit works. The circuit uses a two-qubit gate as the building block, whose repeated utilization in a cascaded fashion yields any general m-party entanglement as required. We then outline the realization of the circuit using optical photon quantum computing.

A. Quantum circuit for entanglement

The entanglement constructed in the previous chapter has a structure dependent on the Hamming weight of basis states. This hints at a cascaded structure for quantum gates. The circuit can be implemented by repeatedly applying a building block U shown in Fig. 4. We make use of the gates which figure frequently in universal quantum computation, namely the controlled-NOT (C-NOT) and Hadamard gates. The gates X and Z are Pauli group members, representing the single-qubits operations of bit-flip and phase-flip respectively.

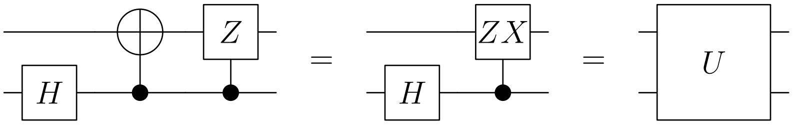

Fig. 4. The building block to form quantum entanglement in (3.2).

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \; Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

We are only interested in creating a specific entanglement (3.2), and thus we only consider a transformation by U that helps this goal. The action of U on the basis state $|11\rangle$ is

$$U(|1\rangle \otimes |1\rangle) = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle).$$

This also happens to be the entanglement for two-party case (though we do not have a protocol for the same).

We build the three-party entanglement as in Fig. 5. As shown above, we refer to an m-party circuit by $U^{(m)}$. The circuit for the three-party case gives the entanglement as

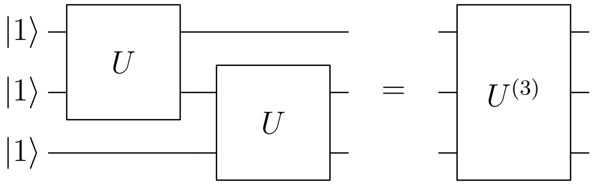

Fig. 5. Quantum circuit for three-party entanglement in [5].

$$U^{(3)}(|1\rangle \otimes |1\rangle \otimes |1\rangle) = \frac{1}{2}(|001\rangle + |010\rangle + |100\rangle - |111\rangle).$$

This also happens to be the entanglement in [5].

For $m \geq 3$, U extends to $U^{(m)}$ recursively, and $U^{(m)}|1\rangle^{\otimes m} = |\varphi\rangle$ of (3.2). This is shown in Fig. 6.

We use induction to prove that the circuit actually forms the entanglement we require in (3.2). We have already proved two base cases, for $m = 2$ and $m = 3$. Next, we prove that whenever the circuit $U^{(m)}$ works, $U^{(m+1)}$ works as well.

Consider a basis state of $|\varphi\rangle$ (3.2),

$$|q\rangle = u(q)|q_j^1 q_j^2 \cdots q_j^m\rangle,$$

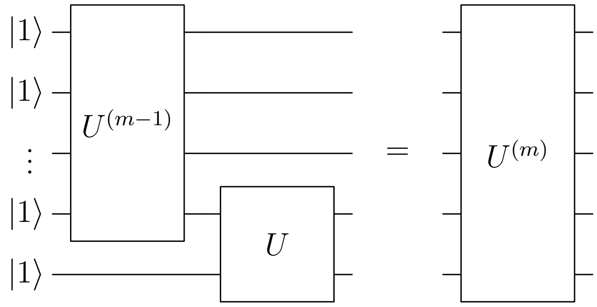

Fig. 6. The recursive application of U to form quantum entanglement in (3.2).

where $q \in \mathbb{F}_2^m$. We consider $|q\rangle$ for which $u(q) = \pm 1$. After $U^{(m)}$, the action of U is

$$\begin{aligned} u(q)|q_j^1 q_j^2 \cdots q_j^m\rangle \otimes |1\rangle &\xmapsto{H} u(q)(|q_j^1 q_j^2 \cdots q_j^m 0\rangle - |q_j^1 q_j^2 \cdots q_j^m 1\rangle) \\ &\xmapsto{C_1-X} u(q)(|q_j^1 q_j^2 \cdots q_j^m 0\rangle - |q_j^1 q_j^2 \cdots \overline{q_j^m} 1\rangle) \\ &\xmapsto{C_1-Z} u(q)(|q_j^1 q_j^2 \cdots q_j^m 0\rangle - (-1)^{\overline{\overline{q_j^m}}} |q_j^1 q_j^2 \cdots \overline{q_j^m} 1\rangle). \end{aligned}$$

Of the two basis states, the first does not undego a change in phase (it remains as $u(q)$), and thus, has the proper sign as per induction hypothesis. The second one is

$$|q'\rangle = -u(q)(-1)^{q_j^m} |q_j^1 q_j^2 \cdots \overline{q_j^m} 1\rangle.$$

The following two cases are possible.

- *Case* $q_j^m = 0$. We get $u(q') = -u(q)$. This is the required result because the Hamming weight of $q_j^1 q_j^2 \cdots \overline{q_j^m} 1$ is two more than that of q.

- *Case* $q_j^m = 1$. We get $u(q') = u(q)$. This is the required result because the Hamming weight of $q_j^1 q_j^2 \cdots \overline{q_j^m} 1$ is equal to that of q.

Thus, $U^{(m+1)}$ works as required, provided that $U^{(m)}$ does.

B. Implementation by optical photon quantum computing

The optical photon has been a prominent candidate for representing a quantum bit in the realization of quantum computing. We take up the optical photon to show our implementation because the respective quantum circuits can be experimentally realized using easily avaiable apparatus such as wave plates and beamsplitters. Furthermore, our basic building block U has a structure which is highly amenable to interactions of optical photons.

1. Introduction to optical photon quantum computing

Optical photons have a number of attractive properties – they are chargeless, they do not interact very strongly with the surrounding matter, and they can be transmitted over long distances by optical fibers without significant loss. As a consequence, they are potentially free from decoherence. However, this strength also becomes a weakness – photons do not interact very strongly with each other. Nevertheless, there exist nonlinear optical media which can facilitate such operations, with the principle of *intereference* being the primary vehicle for such interactions.

a. Encoding qubits into photons

There are a number of ways of representing qubits using photons. Two of the well-known schemes are *dual-rail* representation and *polatization encoding.* While one links the basis states to the spatial presence of photons, the other links them to polarization components.

Light is known to have various *polarization* modes, which refer to the plane in which the photons lie, perpendicular to their propagation path. Light may be horizontally or vertically polarized, in reference to two perpendicular planes. In the polarization mode, the basis states are represented by horizontal and vertical polarization components. Specifically, we may have horizontal component represent $|0\rangle$ and the vertical component $|1\rangle$. Polarizing beamsplitters can be used to separate the two components, as shown in the Fig. 7.

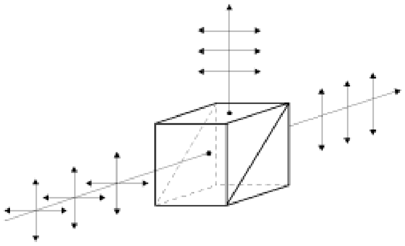

Fig. 7. A polarizing beamsplitter.

In dual-rail representation, the state of a qubit is determined by the presence of a photon in a superposition of two paths. Quantum-mechanically, a photon is in a superposition of two spatially separated paths, one path representing the state $|0\rangle$ and the other $|1\rangle$. Unlike polarization mode which can occur naturally, dual-rail representation needs to be specially prepared. There are many optical setups which can help us prepare the dual-rail encoding from polarization mode. One obvious method is to use the polarizing beamsplitter. As apparent in Fig. 8, the polarizing beam splitter creates two paths for the two polarization components, effectively providing a dual-rail representation. The horizontal and vertical components ($|H\rangle$ and $|V\rangle$)

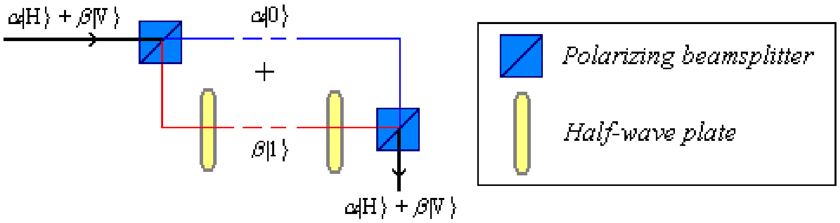

Fig. 8. Conversion of polarization mode to dual-rail and back.

respectively represent the basis states $|0\rangle$ and $|1\rangle$. These states become spatially separated after the action of the beamsplitter. The two slabs labeled 45^o are *half-wave plates* – devices used to compensate for phase shift incurred due to self-interference of the two components at the time of merging.

b. Optical devices for implementing quantum operators

Quantum gates can be realized using some basic optical devices. The primary task in realization is controlled generation of photons. This can be achieved through attenuated laser output. As for operations on photons, the most commonly known device is the mirror, used to change the propagation direction of photons.

Another device is the beamsplitter (shown in Fig. 9), which is typically fabricated from two prisms glued together by a thin metallic base. It act by reflecting a fraction of the incident light, and transmitting the rest. It can also be used to combine incoming beams of light by the principle of interference. Polarizing beamsplitter is a special case of this device, whose splitting operation relies on polarization components.

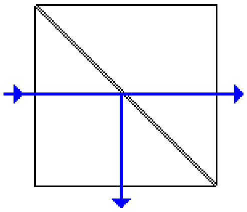

Fig. 9. A schematic depiction of a beamsplitter.

The task of introducing phase shifts can be performed by a wave plate (Fig. 10), which alters the polarization of light by shifting the phase between the perpendicular components. Typically, a wave plate is a birefringent[1] crystal with a thickness chosen

[1] *Birefringence* is the decomposition of light based on its polarization.

depending upon the desired phase shift to be produced. Owing to such action, wave plates are candidates for implementing the Hadamard gate. Specifically, a half-wave plate causes a shift of π – precisely what a Hadamard gate does.

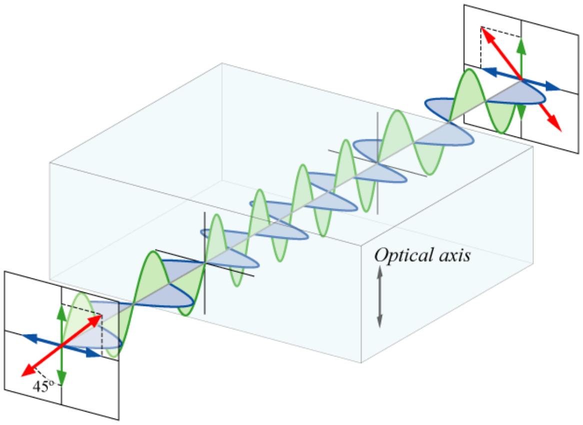

Fig. 10. A wave plate used to shift polarization of photons.

Readers interested in more details of optical photon quantum computing can refer to [4, 8, 13]. An excellent introduction to quantum optics and related concepts is provided in [12].

2. Implementation of quantum entanglement

We now discuss the realization of U that was defined earlier. We realize it using controlled-Z (C-Z) gates and Hadamard gates, which have been studied well for optical photon quantum computing. We make use of the results stated in [11] to outline the physical implementation.

The realization is simplified when we observe that the C-Z gate can be implemented using C-NOT and Hadamard gates. Similarly, a C-NOT can be implemented with C-Z and Hadamard gates. The circuits are shown in Fig. 11.

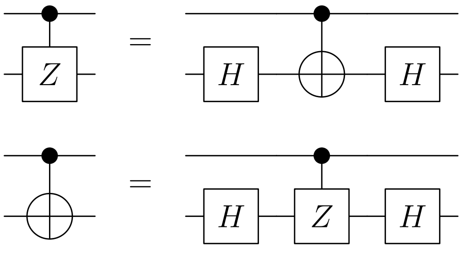

Fig. 11. Implementations of controlled not and phase-flip gates.

a. Realization of the C-NOT gate

In [11], the C-NOT gate is implemented for polarization encoding. However, the internal implementation uses spatial encoding, and polarizing beamsplitters are used for back-and-forth conversions among the two encodings.

Fig. 12 provides a schematic of the internal implementation of the C-NOT gate. Initially, we have the control and target qubits as photons C_{in} and T_{in} respectively. The gate internally uses spatial encoding: the paths C_0, C_1 denote the control qubit, and T_+, T_- denote the target qubit. This spatial encoding is obtained using beamsplitters on C_{in} and T_{in}.

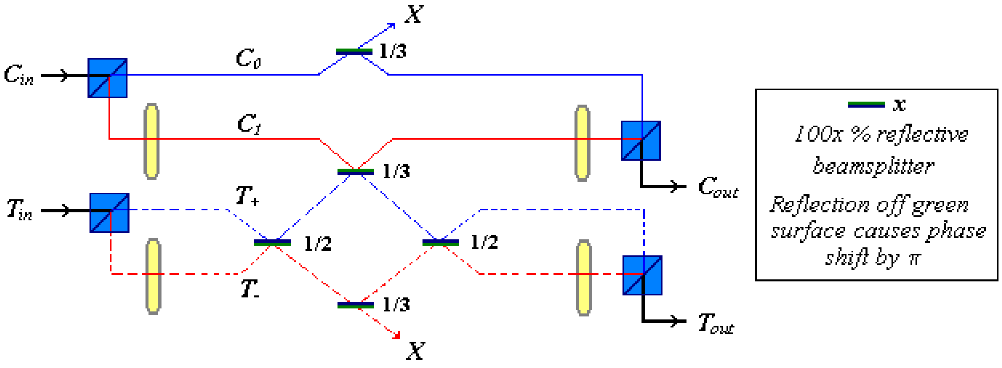
X
C_{in}
C_0
1/3
C_1
1/3
C_{out}
T_{in}
T_+
1/2
1/2
T_-
1/3
T_{out}
X
x
100x % reflective beamsplitter
Reflection off green surface causes phase shift by π

Fig. 12. Schematic of an optical C-NOT gate.

Internally, the green-and-blue plates shown are beamsplitters, which perform the task of either splitting or interfering the incoming path(s). The plates labeled "1/2" act as Hadamard gates[2]. The X on the top and bottom indicate dumped paths; they may be used to determine whether the gate operates as expected. In the context of this implementation, the action of the beamsplitters amounts to creating the desired non-classical interference between photon states, resulting in the quantum mechanical C-NOT action.

In the broad scheme of functioning, the key part of the action of this gate is that the spatial mode encoding causes an interaction among the photons only when the control bit is in the logical state $|1\rangle$. When in path C_0, the photon does not interact with the target, but in case of path C_1, an interaction takes place. The effective action of the interference is to cause a phase flip of the target qubit whenever the control qubit has state $|1\rangle$. With the presence of two Hadamard gates, this effectively becomes the C-NOT operation.

The same setup could be used to implement a C-Z gate, without the Hadamard transforms. The schematic of such a gate would be as in Fig. 13.

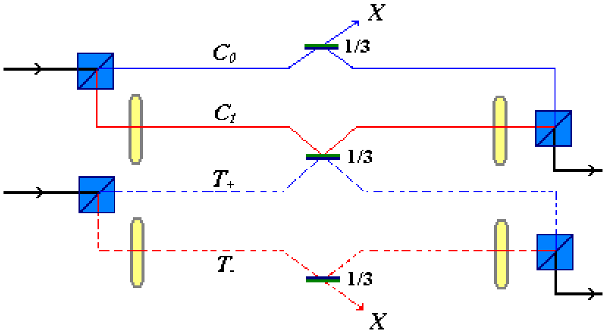

Fig. 13. Schematic of an optical C-Z gate.

[2] The beamsplitters labelled "1/2" are only shown as a part of the schematic. In actual implementation, Hadamard gates are implemented using half-wave plates in polarization encoding, rather than using beamsplitters in spatial encoding.

b. Realization of U

In our implementation, the two basis states $|0\rangle$ and $|1\rangle$ may be respectively represented by horizontal and vertical polarizations. As per the setup in [11], the Hadamard gate can be easily realized using a half-wave plate, and the C-NOT gate effectively makes use of Hadamard and C-Z gates. In these terms, our relevant quantum circuit for U can be realized as in the Fig. 14. The realization of our circuit is complete with coherent photon sources and the setup described in the preceding section. The C-Z gate can be realized by removing the Hadamard gates (half-wave plates) in the C-NOT realization of [11]. The Hadamard gates can be realized by merely placing half-wave plates in the paths of photons representing our qubits. Fig. 15 depicts the realization of our building-block, with reference to the quantum circuit in Fig. 14. The block labeled C-Z precisely stands for Fig. 13.

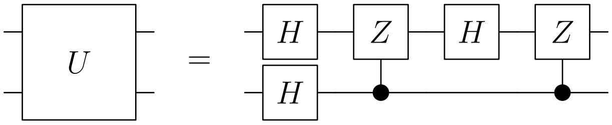

Fig. 14. Building block U realized with H and C-Z gates.

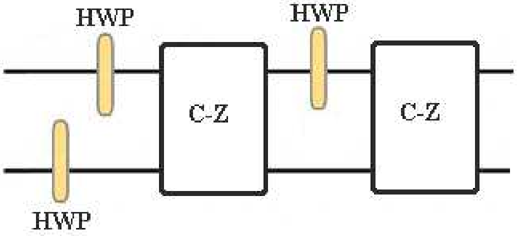

Fig. 15. A schematic of realizing the two-qubit U gate.

CHAPTER V

CONCLUSIONS AND FUTURE WORK

A. Implications to network coding

Network information flow (or network coding) has emerged as an inter-disciplinary field from information theory, coding theory and graph theory, with the goal of maximizing information flow over a network. It does away with the conventional perspective of modeling information as a commodity. A canonical example is the butterfly network (Fig. 16), from the seminal paper [1]. In this example, the goal is for S to send packets x and y to both T_1 and T_2. This can be achieved by single use of the edges if the node C can "combine" the packets such that the T_i's can decode x and y successfully. If data is modeled as a commodity rather than information, C has to transmit twice to achieve the same goal.

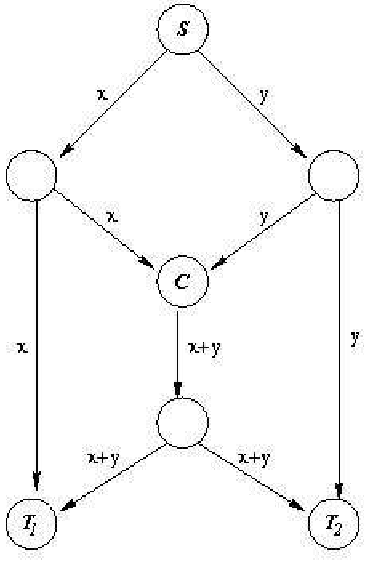

Fig. 16. Multicast transmission in the butterfly network using network coding.

Through an example, we try to investigate the possible impacts of having entanglement-assisted communication in network coding. There have been a number of bounds proven for various network coding settings. For instance, [14] proves a lower bound on the packet size, for successfully performing network coding. It shows that the multicast setting in Fig. 17 (1 source, 6 terminals) requires an alphabet size of at least 3. This bound can be broken if we provide a little power to the network model.

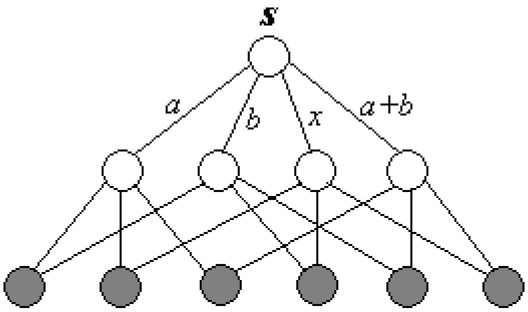

Fig. 17. Multicast transmission requires an alphabet size of at least 3.

We can assume *reverse carpooling* available to one of the links connected to the terminals. Reverse carpooling allows for a special property of the link – both connected nodes simultaneously send a packet from each side of the link, and in turn receive a some function of the two packets sent. Such slight power is able to break the bound on alphabet size[1], and it is now possible to transmit packets of alphabet size 2 by network coding.

Such a power exemplifies the probable impact of providing entanglement assistance to network coding. Provision of entanglement adds power to the existing communication model, and as seen above, a slight usage of such power can lead to overcoming bounds on communication parameters. There are several other impossibility results which can be overcome by reverse carpooling; thus, they could also

[1] This result was discovered during the term project of the network coding course.

provide starting points to investigate the impact of enanglement assistance.

B. Future research directions

In this thesis, we addressed the advantage of using quantum entanglement to reduce the complexity of a certain class of communication problems among multiple parties. We showed the reduction explicitly in case of a four-party communication problem, by presenting a 3-bit protocol that uses entanglement to evaluate a certain function. We also proved that in case entanglement is not present, a classical four-party protocol will require at least 5 bits to evaluate the same function. We further presented a method for formulating similar communication problems for m parties, $m \geq 4$, showing a scalability of the gain in communication complexity. This family of protocols depicts a linear gap in communication complexity when aided by entanglement, as shown by the tight bounds. Our results thus go on to show the utility of entanglement for computation in a distributed setting. As seen in the previous section, we can look for applications of similar protocols in network information flow.

There are a number of open problems associated with our results. For instance, we have a certain structure of entanglement, and we varied the input condition and evaluation function to arrive at a general protocol. One could try to observe the result of reversing these roles. We suspect that in such a case, Protocol Q would no longer be identical for different parties. Moreover, other types of entanglement may be explored to help reduce classical communication complexities. Another general direction of pursuit is exploration of super-linear gains in communication complexity, since the gain we have shown is linear. Also, our results could be actually converted to bandwidth gains in practical settings.

There are a number of possible ways to improve upon our protocols. For instance, we could investigate the existance of an entanglement which can be reused for different indices of the input bit-strings, thus resulting in a total of m qubits used instead of mn. Such approach may be investigated by increasing some of the input preconditions and observing the impact on communication complexity. In general, an obvious improvement would be the reduction in quantum computing resources used. It can be seen that quantum entanglement is an additional resource, and it does not have an analogous "resource" in the classical protocols. An optical photon quantum computing realization of this protocol, as explained in the previous chapter, could help account for the cost factors involved, in order to get a more stringent notion of fairness in our comparison.

REFERENCES

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, “Network information flow,” *IEEE Transactions on Information Theory*, vol. 46, pp. 1204–1216, July 2000.

[2] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters, “Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels,” *Phys. Rev. Lett.*, vol. 70, pp. 1895–1899, Mar 1993.

[3] H. Buhrman, R. Cleve, and W. van Dam, “Quantum entanglement and communication complexity,” *arXiv:quant-ph/9705033*, 1997.

[4] I. L. Chuang and Y. Yamamoto, “Simple quantum computer,” *Phys. Rev. A*, vol. 52, pp. 3489–3496, Nov 1995.

[5] R. Cleve and H. Buhrman, “Substituting quantum entanglement for communication,” *Phys. Rev. A*, vol. 56, pp. 1201–1204, Aug 1997.

[6] A. Holevo, “Information-theoretical aspects of quantum measurement,” *Problemy Peredachi Informatsii*, vol. 9, no. 2, pp. 31–42, 1973.

[7] G. Hutchinson, “Partioning algorithms for finite sets,” *Commun. ACM*, vol. 6, no. 10, pp. 613–614, 1963.

[8] P. Kok, W. J. Munro, K. Nemoto, T. C. Ralph, J. P. Dowling, and G. J. Milburn, “Linear optical quantum computing,” *arXiv:quant-ph/0512071v2*, 2005.

[9] E. Kushilevitz and N. Nisan, *Communication Complexity.* Cambridge, U.K.: Cambridge University Press, 1997.

[10] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information.* Cambridge, U.K.: Cambridge University Press, 2000.

[11] J. L. O'Brien, G. J. Pryde, A. G. White, T. C. Ralph, and D. Branning, "Demonstration of an all-optical quantum controlled-NOT gate," *Nature*, vol. 426, pp. 264–267, Nov. 2003.

[12] H. Paul, *Introduction to Quantum Optics: From Light Quanta to Quantum Teleportation*. Cambridge, U.K.: Cambridge University Press, 2004.

[13] T. C. Ralph, A. G. White, W. J. Munro, and G. J. Milburn, "Simple scheme for efficient linear optics quantum gates," *Phys. Rev. A*, vol. 65, pp. 012314–1 – 012314–6, Dec 2001.

[14] A. Rasala Lehman and E. Lehman, "Complexity classification of network information flow problems," in *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, New Orleans, LA, January 2004, pp. 142–150.

[15] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Rev.*, vol. 41, no. 2, pp. 303–332, 1999.

[16] A. M. Turing, "On computable numbers, with an application to the entscheidungsproblem," *Proc. London Math. Soc.*, vol. 2, no. 42, pp. 230–265, 1936.

[17] A. C.-C. Yao, "Some complexity questions related to distributive computing(Preliminary Report)," in *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, Atlanta, GA, April-May 1979, pp. 209–213.

APPENDIX A

DISTRIBUTION OF ORTHOGONAL SUBSPACES

Let B^0 denote the sub-space orthogonal to $b \in \mathbb{F}_2^\ell$, and let $B^1 = \mathbb{F}_2^\ell \setminus B^0$. Also, let $B_{i,j}^k$ denote the respective sets with each member r satisfying the condition $|r|_j = i$.

$$B_{i,j}^k = \{r : r \in B^k, |r|_j = i\}.$$

For the remainder of this section, we assume $k \in \mathbb{Z}_2$ and $x \in \mathbb{Z}_4$. It is known that $|B^k| = 2^{\ell-1}$. We wish to determine $|B_{x,4}^k|$.

We have the following simple observation.

Lemma 7. *Let $b, b' \in \mathbb{F}_2^\ell$, and let r be the number of common 1's between them. Then, $|b + b'| = |b| + |b'| - 2r$, and $b' \perp b \Leftrightarrow r \equiv 0 \bmod 2$.*

We know $\mathbb{F}_2^\ell$ contains $\binom{\ell}{k}$ vectors such that $|q| = k$. Thus, $\sum \binom{\ell}{4k+x}$ gives the number of vectors q such that $|q|_4 \equiv x \bmod 4$. We have [1]

$$\sum_{k=0}^{\lfloor \frac{\ell}{4} \rfloor} \binom{\ell}{4k+x} = 2^{\ell-2} + 2^{\frac{\ell}{2}-1} \cos(\frac{\ell - 2x}{4}\pi). \tag{A.1}$$

When $b \in \mathbb{F}_2^\ell \setminus \{\hat{0}, \hat{1}\}$, we have the following arguments.

1. Consider any vector $b' \in B_{1,2}^0$ such that $|b'|_2 = 1$. For any $b'' \in B_{0,2}^0$, $b' + b'' \in B_{1,2}^0$. Thus, $|B_{0,2}^0| = |B_{1,2}^0| = 2^{\ell-2}$.

2. Consider any vector $b' \in B_{1,2}^1$ such that $|b'|_2 = 1$. For any $b'' \in B_{0,2}^0$, $b' + b'' \in B_{1,2}^1$. Also, for any $b'' \in B_{1,2}^0$, $b' + b'' \in B_{0,2}^1$. Thus, $|B_{0,2}^1| = |B_{1,2}^1| = 2^{\ell-2}$.

[1] We can easily prove (A.1) by the binomial expansion of $(1 + i)^\ell = \sum_{j=0}^{\ell} \binom{\ell}{j} i^j$ (where $i = \sqrt{-1}$), along with the fact that $\sum \binom{\ell}{2k} = \sum \binom{\ell}{2k+1} = 2^{\ell-1}$.

3. $|b|_2 = 0$. Clearly, $b \in B^0$. Also, $\hat{1} \in B^0$.

   (a) $\ell \equiv 0 \bmod 4$.

      i. Consider $B^0_{1,2}$ as $B^0_{1,4} \cup B^0_{3,4}$. For any $b' \in B^0_{1,2}$, $b' + \hat{1} \in B^0_{1,2}$. But if $b' \in B^0_{1,4}$, then $b' + \hat{1} \in B^0_{3,4}$ due to the fact that $\ell \equiv 0 \bmod 4$. Thus, $|B^0_{1,4}| = |B^0_{3,4}| = 2^{\ell-3}$.

      ii. Consider $B^1_{1,2}$ as $B^1_{1,4} \cup B^1_{3,4}$. For any $b' \in B^1_{1,2}$, $b' + \hat{1} \in B^1_{1,2}$. But if $b' \in B^1_{1,4}$, then $b' + \hat{1} \in B^1_{3,4}$ due to the fact that $\ell \equiv 0 \bmod 4$. Thus, $|B^1_{1,4}| = |B^1_{3,4}| = 2^{\ell-3}$.

      iii. We have the following cases.

         A. $|b|_4 = 2$. Consider $B^0_{0,2}$ as $B^0_{0,4} \cup B^0_{2,4}$. For any $b' \in B^0_{0,2}$, $b' + b \in B^0_{0,2}$. But if $b' \in B^0_{2,4}$, then $b' + b \in B^0_{0,4}$ by Lemma 7. Thus, $|B^0_{0,4}| = |B^0_{2,4}| = 2^{\ell-3}$.

         B. $|b|_4 = 0$. Consider $B^1_{0,2}$ as $B^1_{0,4} \cup B^1_{2,4}$. For any $b' \in B^1_{0,2}$, $b' + b \in B^1_{0,2}$. But if $b' \in B^1_{0,4}$, then $b' + b \in B^1_{2,4}$ by Lemma 7. Thus, $|B^1_{0,4}| = |B^1_{2,4}| = 2^{\ell-3}$.

   (b) $\ell \equiv 2 \bmod 4$.

      i. Consider $B^0_{0,2}$ as $B^0_{0,4} \cup B^0_{2,4}$. For any $b' \in B^0_{0,2}$, $b' + \hat{1} \in B^0_{0,2}$. But if $b' \in B^0_{0,4}$, then $b' + \hat{1} \in B^0_{2,4}$ due to the fact that $\ell \equiv 2 \bmod 4$. Thus, $|B^0_{0,4}| = |B^0_{2,4}| = 2^{\ell-3}$.

      ii. Consider $B^1_{0,2}$ as $B^1_{0,4} \cup B^1_{2,4}$. For any $b' \in B^1_{0,2}$, $b' + \hat{1} \in B^1_{0,2}$. But if $b' \in B^1_{0,4}$, then $b' + \hat{1} \in B^1_{2,4}$ due to the fact that $\ell \equiv 2 \bmod 4$. Thus, $|B^1_{0,4}| = |B^1_{2,4}| = 2^{\ell-3}$.

      iii. We have the following cases.

         A. $|b|_4 = 2$. Consider $B^0_{1,2}$ as $B^0_{1,4} \cup B^0_{3,4}$. For any $b' \in B^0_{1,2}$, $b' + b \in$

$B^0_{1,2}$. But if $b' \in B^0_{1,4}$, then $b' + b \in B^0_{3,4}$ by Lemma 7. Thus, $|B^0_{1,4}| = |B^0_{3,4}| = 2^{\ell-3}$.

B. $|b|_4 = 0$. Consider $B^1_{1,2}$ as $B^1_{1,4} \cup B^1_{3,4}$. For any $b' \in B^1_{1,2}$, $b' + b \in B^1_{1,2}$. But if $b' \in B^1_{1,4}$, then $b' + b \in B^1_{3,4}$ by Lemma 7. Thus, $|B^1_{1,4}| = |B^1_{3,4}| = 2^{\ell-3}$.

4. $|b|_2 = 1$. Clearly, $b \in B$. Also, $\hat{1} \in B$.

(a) For any $b' \in B^0_{0,2}$, $b' + b \in B^1_{1,2}$. We have the following specific cases.

i. $|b|_4 = 1$. For any $b' \in B^0_{0,4}$, $b' + b \in B^1_{1,4}$ by Lemma 7. Also, if $b' \in B^0_{2,4}$, then $b' + b \in B^1_{3,4}$ by Lemma 7. Thus, $|B^0_{0,4}| = |B^1_{1,4}|$ and $|B^0_{2,4}| = |B^1_{3,4}|$.

ii. $|b|_4 = 3$. For any $b' \in B^0_{0,4}$, $b' + b \in B^1_{3,4}$ by Lemma 7. Also, if $b' \in B^0_{2,4}$, then $b' + b \in B^1_{1,4}$ by Lemma 7. Thus, $|B^0_{0,4}| = |B^1_{3,4}|$ and $|B^0_{2,4}| = |B^1_{1,4}|$.

(b) $\ell \equiv 0 \bmod 4$.

i. For any $b' \in B^0_{0,4}$, $b' + \hat{1} \in B^1_{0,4}$; also, for any $b' \in B^0_{2,4}$, then $b' + \hat{1} \in B^1_{2,4}$ – all due to the fact that $\ell \equiv 0 \bmod 4$. Thus, $|B^0_{0,4}| = |B^1_{0,4}|$ and $|B^0_{2,4}| = |B^1_{2,4}|$.

ii. For any $b' \in B^0_{1,4}$, $b' + \hat{1} \in B^1_{3,4}$; also, for any $b' \in B^0_{3,4}$, then $b' + \hat{1} \in B^1_{1,4}$ – all due to the fact that $\ell \equiv 0 \bmod 4$. Thus, $|B^0_{1,4}| = |B^1_{3,4}|$ and $|B^0_{3,4}| = |B^1_{1,4}|$.

(c) $\ell \equiv 2 \bmod 4$.

i. For any $b' \in B^0_{1,4}$, $b' + \hat{1} \in B^1_{1,4}$; also, for any $b' \in B^0_{3,4}$, then $b' + \hat{1} \in B^1_{3,4}$ – all due to the fact that $\ell \equiv 2 \bmod 4$. Thus, $|B^0_{1,4}| = |B^1_{1,4}|$ and $|B^0_{3,4}| = |B^1_{3,4}|$.

ii. For any $b' \in B^0_{0,4}$, $b' + \hat{1} \in B^1_{2,4}$; also, for any $b' \in B^0_{2,4}$, then $b' + \hat{1} \in B^1_{0,4}$ – all due to the fact that $\ell \equiv 2 \bmod 4$. Thus, $|B^0_{0,4}| = |B^1_{2,4}|$ and

$$|B^0_{2,4}| = |B^1_{0,4}|.$$

The above analysis gives us a system of equations for the variables $|B^k_{x,4}|$, for cases $\ell \equiv 0 \pmod 4$ and $\ell \equiv 2 \pmod 4$. After solving with (A.1), we get the following lemma. It puts the valued of $|B^k_{x,4}|$ in a general form in terms of k and x.

Lemma 8. *For $b \in \mathbb{F}_2^\ell \setminus \{0, 1\}$,*

1. $\ell \equiv 0 \bmod 4 \Rightarrow$

$$|B^k_{x,4}| = 2^{\ell-3} + 2^{\frac{\ell}{2}-2}(-1)^{\frac{\ell}{4}} \left(\cos(\frac{x}{2}\pi) + (-1)^k \cos(\frac{x + |b|_4}{2}\pi) \right).$$

2. $\ell \equiv 2 \bmod 4 \Rightarrow$

$$|B^k_{x,4}| = 2^{\ell-3} + 2^{\frac{\ell}{2}-2}(-1)^{\frac{\ell-2}{4}} \left(\sin(\frac{x}{2}\pi) + (-1)^k \sin(\frac{x + |b|_4}{2}\pi) \right).$$

When $b \in \{\hat{0}, \hat{1}\}$, we have the following lemma.

Lemma 9. *For $b \in \{\hat{0}, \hat{1}\} \subseteq \mathbb{F}_2^\ell$,*

1. *When $b = \hat{0}$, $|B^0_{x,4}| = \sum \binom{\ell}{4k+x}$ and $|B^1_{x,4}| = 0$.*
2. *When $b = \hat{1}$, $B^0_{1,2} = B^1_{0,2} = \{\}$, $|B^0_{0,4}| = \sum \binom{\ell}{4k}$, $|B^1_{1,4}| = \sum \binom{\ell}{4k+1}$, $|B^0_{2,4}| = \sum \binom{\ell}{4k+2}$, and $|B_{3,4}| = \sum \binom{\ell}{4k+3}$.*

Proof. 1. $b = \hat{0}$. Clearly, $B^0 = \mathbb{F}_2^\ell$. Result follows from (A.1).

2. $b = \hat{1}$. Since $\ell \equiv 0 \bmod 2$, $r \cdot b = |r|_2$. We get $B^0_{0,2} = \{r : r \in \mathbb{F}_2^\ell,\ |r|_2 = 0\}$, and $B^1_{1,2} = \{r : r \in \mathbb{F}_2^\ell,\ |r|_2 = 1\}$.

□

APPENDIX B

PROGRAM FOR PROOF OF LOWER BOUND

The following is the code for two files - partition.h and lb.C; both were written to prove that no 4-bit communication protocol exists without using quantum entanglement. The file partition.h implements the partition-generating algorithm proposed in [7]. The program in lb.C exaustively examines every combination of four partitions from the four parties, and finds the valid inputs that result in ambiguous function values. The code is written in C++. The expected output is the string "No protocol can solve the current problem.".

```
/* partitions.h - contains definition of the Partitions class
 *  The class will be used in lb.C to prove lower bound */


#ifndef __PARTITIONS_H
#define __PARTITIONS_H


#include <iostream>
#include <vector>


#define PRINT_A 1000
#define PRINT_SETS 1001
#define PRINT_NUM 1002


using namespace::std;
```

```
class Partitions {
  private:
    vector <int> a;
    vector <int> c;
    int n, m, I, J, C, K;
    int setCount;
    bool started;
  public:
    Partitions(int nInit=0, int mInit=0);
    void init();
    bool getNext();
    void print(int method=PRINT_A);
    int ownerSetOf(int e);
};

Partitions::Partitions(int nInit, int mInit) {
  if( nInit < mInit || nInit < 0 || mInit < 0 )
    cout << "Error in set paramters" << endl ;
  else {
    n = nInit; m = mInit;
    init();
  }
}

void Partitions::init() {
  c.clear(); a.clear();
  for(int i=0; i<m; i++) {
    c.push_back(i);
    a.push_back(i);
  }
  for(int i=m; i<n; i++)
    a.push_back(0);
  c.push_back(0);
  I = J = 0;
  started = false;
  setCount = 0;
}

void Partitions::print(int method) {
  if( method == PRINT_A ) {
    for(int i=0; i<n; i++)
      cout << a[i] << " ";
  }
  else if( method == PRINT_NUM ) {
    cout << setCount ;
    return ;
  }
  else {
    for(int i=0; i<m; i++) {
      for(int j=0; j<n; j++)
        if( a[j] == i )
          cout << j << " ";
      cout << "\t\t";
    }
  }
  cout << endl;
}

int Partitions::ownerSetOf( int e ) {
  if( e < 0 || e >= n )
    return -1;

  return a[e];
}

/* Implementation of the algorithm proposed
 *   by Hutchinson in 1963 (see references) */
bool Partitions::getNext() {

  if(started) goto six ;
  else started = true;

three:
  if( c[J] == I ) J++;
  else a[I] = 0;
four:
  if( I == n-1 ) goto five;
  else if( I < n-1 ) { I++; goto three; }
```

```
five:
  setCount++;
  return true; /* A new partition set has been created */
six:
  I = n-1; J = m-1;
seven:
  if( c[J] == I ) { J--; goto nine; }
  else goto eight;
eight:
  if( a[I] < J ) goto ten;
  else if( a[I] >= J ) goto nine;
nine:
  I--;
  if( I > 0 ) goto seven;
  else if( I == 0 ) goto thirteen;
ten:
  a[I]++; J++;
eleven:
  if( I == n-1 ) goto five;
  else if( I < n-1 ) { I++; goto twelve; }
twelve:
  if( c[J] == I ) { a[I] = J; J++; goto eleven; }
  else { a[I] = 0; goto eleven; }
thirteen:
  C = n-1; K = m-1;
fourteen:
  if( c[K] < C ) { c[K]++; goto sixteen; }
  else if( c[K] >= C ) goto fifteen;
fifteen:
  C = c[K] - 1; K--;
  if( K == 0 ) return false; /* No more partitions can be created */
  if( K > 0 ) goto fourteen;
sixteen:
  if( K == m-1 ) { I = J = 0; goto three; }
  else if( K < m-1 ) { K++; goto seventeen; }
seventeen:
  c[K] = c[K-1] + 1; goto sixteen;

}

#endif

/**********************************************************/

/* lb.C - program that performs exhaustive search among
 *  partitioning strategies to look for a protocol.
 *  This particular program fails to provide any protocol,
 *  thereby proving the lower bound. */

#include <iostream>
#include "partitions.h"
```

```
#define INITIAL 100

#define is_valid(a,b,c,d) ( (((a)+(b)+(c)+(d))%2) == 0 )
/* Whether the four input strings are valid */
#define computeFunction(a,b,c,d) ( ( ((a)+(b)+(c)+(d))>>1 ) & 0x1  )
/* Computes the four-argument function f(xA, xB, xC, xD) */

int main() {
  Partitions pBob(4,4), pCarol(4,2), pDan(4,2);
   // partitions used by the three parties
   // First argument to constructor gives size of the set,
   //  and second argument gives number of partitions to
   //  be created from the given set.
  int originalFunctionValue[4*16], f;
   // array storing (possibly) ambiguous function values
  int originalFunctionArguments[4*16][3];
   // arguments of original function value
  int index; // control and index variables, respectively
  int w, x, y, z; // inputs
  bool printFlag = false ;
   // flag to know whether failure needs to be printed
  char in ;

  cout << "Do you want me to print every failure (y/n)? " << endl ;
  cin >> in ;
  if( in == ’y’ ) printFlag = true;
  /* Loop over all partitions for Bob, Carol, Dan and Alice (self).
   *   getNext() generates the next partition. */
  while( pBob.getNext() ) {
    pCarol.init();
    while( pCarol.getNext() ) {
      pDan.init();
      while( pDan.getNext() ) {
for(int i=0; i<4*16; i++)
  originalFunctionValue[i] = INITIAL;

/* Running a loop representing all possible input combinations */
for(int input=0; input<256; input++) {
  /* Extract inputs from the entire bit-string */
  w = input & 0x3;  z = (input>>2) & 0x3;
  y = (input>>4) & 0x3; x = (input>>6) & 0x3;
  if( is_valid(w,x,y,z) ) { // Process only valid inputs
    f = computeFunction(w,x,y,z);
    /* indexing by unique input of Alice and various
     *   partition numbers */
      index = w<<4 | pBob.ownerSetOf(x)<<2
        | pCarol.ownerSetOf(y)<<1
        | pDan.ownerSetOf(z);
    /* The first time a partition is examined */
            if( originalFunctionValue[index] == INITIAL ) {
      originalFunctionValue[index] = f;
      originalFunctionArguments[index][0] = x;
      originalFunctionArguments[index][1] = y;
      originalFunctionArguments[index][2] = z;
    }
    else {
      if( originalFunctionValue[index] != f ) {
        /* Ambiguous function value found, current
         *   partitioning strategy fails */
if( printFlag ) {
   /* We now print what function values it failed for */
  pBob.print(PRINT_NUM);
  pCarol.print(PRINT_NUM);
  pDan.print(PRINT_NUM);
  cout << "(" << w << originalFunctionArguments[index][0]
     << originalFunctionArguments[index][1]
        << originalFunctionArguments[index][2]
        << ")(" << w << x << y << z << "); f=" << f << endl;
}
        goto protocol_failure;
      }
    }
  }
  }
  /* If we reach here, there exists a protocol, since no
   *   conflicting function values were found. */
cout << "We have a protocol. Dumping partitions:" << endl;
```

```
pBob.print(PRINT_SETS);
pCarol.print(PRINT_SETS);
pDan.print(PRINT_SETS);
return 0; /* We already have a protocol, so abandon
 *  any more computations and return. */
        protocol_failure:
         /* Do nothing: reaching here implies current partitioning
          *  strategy has failed; so move on to the next one. */ ;
      }
    }
  }

  cout << "No protocol can solve the current problem." << endl;
  return 0;
}
```

VITA

Name:	Angad Mohandas Kamat
Permanent Address:	“Manas”, House no. 169/B-7 Shantinagar Ponda - Goa, 403401 India
Email address:	kamat.angad@gmail.com
Education	B.Tech., Computer Science and Engineering, National Institute of Technology, Warangal, India, 2006 M.S., Computer Science, Texas A&M University, 2008