LOWER BOUND FOR SCALABLE BYZANTINE AGREEMENT

by

Dan Holtby
B.Sc. University of Victoria 2004

A Thesis Submitted in Partial Fullfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in the Department of Computer Science

Lower Bound for Scalable Byzantine Agreement

by

Dan Holtby

B.Sc. University of Victoria 2004

Supervisory Committee

Dr. Bruce Kapron	(Computer Science)
Supervisor	
Dr. Valerie King	(Computer Science)
Supervisor	
Dr. Venkatesh Srinivasan	(Computer Science)
Departmental Member	
Dr. Anthony Quas	(Mathematics)
External	

Supervisory Committee

Supervisor:	Dr. Bruce Kapron
Supervisor:	Dr. Valerie King
Departmental Member:	Dr. Venkatesh Srinivasan
External:	Dr. Anthony Quas

Abstract

We consider the problem of computing Byzantine Agreement in a synchronous network with n processors each with a private random string, where each pair of processors is connected by a private communication line. The adversary is malicious and non-adaptive, i.e., it must choose the processors to corrupt at the start of the algorithm. Byzantine Agreement is known to be computable in this model in an expected constant number of rounds. We consider a scalable model where in each round each uncorrupt processor can send to any set of $\log n$ other processors and listen to any set of $\log n$ processors. We define the *loss* of a computation to be the number of uncorrupt processors whose output does not agree with the output of the majority of uncorrupt processors. We show that if there are t corrupt processors, then any randomised protocol which has probability at least $1/2 + 1/\log n$ of loss less than $\frac{t^{2/3}}{16fn^{1/3}\log^{5/3} n}$ requires at least f rounds.

Table of Contents

Chapter 1

Introduction

This thesis starts with the premise that for large distributed networks, it is not possible for a processor to communicate directly with all of the other processors. We examine the limits on computing the fundamental problem of Byzantine Agreement in a synchronous network model in which there is a private communication line between each pair of processors, but in each round each processor may send messages to a set of only $\log n$ processors and receive messages from a set of only $\log n$ processors, where n is the total number of processors. Each processor may choose these sets at each step as a function of the messages it has seen so far and its private string of random bits. We assume a malicious adversary who may corrupt t processors. The adversary may coordinate the actions of these corrupt processors, and is able to make them behave in an arbitrary manner. However the adversary is non-adaptive

in the sense that it must decide at the start of the algorithm which processors it will corrupt. Furthermore the corrupt processors are not privy to communications between uncorrupt processors and do not know the private random bit strings. In other words, the only messages that adversary can base its decisions on are those messages that have been received by the corrupt processors. Using the terms defined previously, our model uses synchronous, point-to-point, private communication, and the algorithm faces a non-adaptive, arbitrary-failure adversary. We define the loss of an execution of the protocol to be the number of processors whose output bits disagree with the bits output by the majority. A protocol with 0 loss is the most desirable, however such a protocol may not always be possible, depending on other constraints.

Under our model, we prove a tradeoff between the number of rounds and the amount of loss, and show how this tradeoff varies with t. In particular, if t is a constant fraction of the total number of processors n, then $\tilde{\Omega}(n^{1/3})$ rounds are required for 0 loss. Our result holds even given an initial assignment in which all but $\Theta(n^{1/3})$ uncorrupt processors are given the same bit.

1.1 Motivation

Large scale distributed computing networks are becoming popular for performing non-trivial computations on large amounts of data, as well as for simply sharing

large amounts of data. Gnutella [10] is a peer-to-peer network used for the searching and sharing of files online. Although the ephemeral nature of any network between home computers makes counting difficult, the Gnutella network is estimated to have between 500,000 to over 1,000,000 nodes at any given moment. Another large network is the hybrid peer-to-peer network maintained by Cloudmark [3], that detects spam by comparing fingerprints generated from e-mails marked as spam by other users, and draws its data from over 100 million clients world wide. Both of these networks contain so many nodes that any protocol requiring a linear number of messages to be sent or processed by each node would be completely impractical. The Cloudmark network is not truly peer-to-peer, as admission control and vote-tallying is handled by dedicated servers run by Couldmark. Additionally, admission is fee-based, and as such, one would expect a far lower number of malicious users than in a network with completely open admission. Nevertheless, the network shows a need for large scale networks with many millions of nodes to perform such tasks.

In both of the above networks, malicious users have a keen interest in disrupting the network, or at least manipulating the results. For example, the music industry has been *poisoning* [2] peer-to-peer file sharing networks by injecting fake files with similar lengths and other meta-data, but which are actually corrupt or entirely different than expected, in order to prevent people from downloading files over which they hold the copyright. In the case of the Cloudmark network, one would likewise expect the

multi-billion dollar spam industry to have an immense interest in preventing their own e-mails from being marked as spam, or causing enough good e-mails to be marked as spam as to render the network useless. As mentioned, this is not an issue with the Cloudmark network itself, as it has strict admission controls and trusted, uncorrupt servers performing much of the computation. However, for a true peer-to-peer version of this application, this sort of malicious attack would be of great concern.

Besides these two examples, where malicious users have a clear benefit to attacking the network, one could argue that any distributed network of this sort will almost certainly be attacked in such a way. Two fairly well known examples are Seti@Home [16] and Folding@Home [9], networks charged with searching for Aliens, and simulating the folding of protein sequences, respectively. These networks consist of hundreds of thousands of personal computers performing computationally intensive tasks on large blocks of data sent to them by central servers. Membership is completely voluntary in either network, and participation grants no rewards beyond the lists of top CPU time contributors maintained on the respective websites of the two networks. In spite of this, groups of people have been modifying the client software to return fake data with no computational effort, thus allowing them to artificially inflate their participation numbers. Clearly, if a noticeable proportion of users are willing to interfere with scientific computations searching for cures to cancer and other diseases, simply for the sake of interference, one should expect a large number of malicious users in

any open network, regardless of the purpose of that network.

Byzantine Agreement is one of the most widely studied problems in distributed computing, as it is a simple, fundamental problem, and forms the basis of many secure protocols to solve more complicated problems. As argued above, these is a need for robust, scalable protocols that operate in peer-to-peer networks with millions of nodes. As Byzantine Agreement will be in some way connected to most of such protocols, whether as a component or simply as a related problem, it follows that a scalable protocol for Byzantine Agreement is a worthwhile protocol to investigate.

1.2 Overview

Chapter 2 gives a description of the Byzantine Agreement problem. It then summarises results that are relevant to our work.

Chapter 3 contains a detailed description of our model of computation.

Chapter 4 presents the details of the adversary strategy, as well as the proof that this adversary will give us the desired lower bound. This chapter is the major contribution of this Thesis.

Chapter 5 concludes the thesis and presents possible avenues of further work on the subject of Byzantine Agreement in a scalable model.

Chapter 2

Background

The problem of Byzantine Agreement was first introduced by name by Lamport, Shostak, and Pease [12]. The problem is that the Byzantine army, led by numerous generals and commanded by one supreme commander, is engaged in a battle. The generals are spread out and can only communicate through messengers, and some of the generals may be traitors. The supreme commander will pass orders to the generals, to either attack or retreat. As long as all of the loyal generals follow the same course of action, the army will be victorious. However, it is possible that they have not all received the same orders, due to the commander himself, or his messengers, being traitors. Thus, the generals must decide amongst themselves what course of action to take. The traitors will try to confuse the loyal generals by sending misleading messages, so the system the generals use must be resistant to this sort

of tampering. For the case that the commander and messengers are not corrupt, the generals must always obey their commander. One can also view the problem without the commander, in which case the initial orders of each general represent their assessment of the best course of action. Again, if the loyal generals are initially in agreement on the best course of action, this must be the action they eventually agree upon.

This problem can be extended to distributed computing as follows. There are n processors, t of which are corrupt and being controlled by a single adversary. These processors all have some initial input of a single bit. The non-corrupt processors must communicate with each other in some way, and agree upon a single output value, while the corrupt processors will be attempting to prevent such an agreement. For this agreement to have meaning, it must be the case that output that the uncorrupt processors agree upon was held by at least one of them initially. Whether or not this problem is solvable depends on how the processors can communicate, how they are allowed to make decisions, and how powerful the adversary is allowed to be.

These results shed some light on recent work in scalable distributed computing in an even stronger model of computation. In [11], King, et.al. presented a $\log n / \log \log n$ round scalable protocol for the Leader Election problem and Byzantine Agreement with loss $n / \log n$. This paper left open the question of whether any loss was necessary in a protocol involving polylogarthmic bits of communication per

processor. In that paper, each uncorrupt processor is limited to sending polylogarithmic number of bits per round. While there is no explicit limit on the number of processors that each processor may listen to per round each processor is required to "process" no more than polylogarithmic messages per round. In the protocols given there, each processor knows ahead of time which processors are legitimately sending to it and can specify the polylogarithmic number of other processors which it is willing to receive messages from, as required in our model. Also as in our model, the adversary is malicious and non-adaptive in the sense that it must choose which processors are corrupt at the start and corrupt processors may send an arbitrary number of messages. Under their model, the adversary has access to all messages sent (though not the private random strings. Also note that their bound still holds even with private messages), and is allowed to send messages after the other processors in a given round. We are able to prove our lower bounds using a weaker adversary, i.e. one which does not have access to communications between uncorrupt processors and allowing messages of arbitrary length. As an adversary under a strong model can act like an adversary under a weaker model by not taking advantage of the additional power allowed it, our lower bounds apply to the model in [11], at least as it is employed there.

The model in [11] is a point-to-point version of the full information model used in several leader election papers (e.g., See Feige [5], Russell and Zuckerman [15].) Those

papers are concerned with reducing the number of rounds, assume broadcast by every processor in each round and allow no loss. All these papers assume a randomised protocol as it has long been known that a deterministic protocol must take $t + 1$ rounds to compute Byzantine Agreement, as originally shown by Fischer and Lynch [8]. It is not hard to show that the same result implies that t rounds are needed to compute even if a significant loss is allowed. Micali and Feldman [6] gave a protocol to compute Byzantine Agreement (with no loss) in constant expected number of rounds with $t < n/3$ in a model similar to ours but with an even stronger adversary, i.e., one which is adaptive. It is then striking then to see how imposing the scalability requirement increases the number of rounds needed to compute with no loss. The problem of Byzantine Agreement with loss ("almost everywhere agreement") was addressed by Dwork, et.al. [4]. This paper offers protocols for networks of bounded degree. In such networks of degree d there is an obvious lower bound on the loss in that the corrupt processors can immediately isolate t/d processors. The protocols in that paper are deterministic and therefore necessarily require large numbers of rounds. Lower bounds on the number of rounds to compute Byzantine Agreement with an adaptive fail-stop adversary who can see the random bits of each processor (in a non-scalable model) was shown by Bar-Joseph and Ben-Or [1]. Lewis and Saia [13] give a scalable protocol for computing Byzantine Agreement in a network where processors join at varying times. There, the adversary is unable to corrupt arbitrary

processors but can determine the time at which the corrupt processors enter. Lower bounds on the number of rounds for leader election in non-scalable full information model can be found in Russell et. al. [14]. Fich and Ruppert [7] give an extensive survey of lower bounds in distributed computing.

Chapter 3

Model

We work in a *scalable* synchronous model for distributed computation. There is a set $N = \{1, 2, \ldots, n\}$ of *processors*. Initially, each processor has a single bit of input, given via an *initial assignment*, and a (private) random bitstring. Computation proceeds in *rounds*. At the start of each round, every processor has an *internal state*, which consists of its input, random bits, all messages that it has received so far, as well as the current round. Each processor has a function r of its state which determines its actions in a given round. Each round has three distinct *phases*, which we will view has happening sequentially.

The phases of a round for a given processor i are as follows:

1. Choose a set of $\lg n$ processors from N to listen to.

2. Choose a set of $\lg n$ processors from N to send to, and send (possibly different)

messages to each of these.

3. Update internal state, by including all messages sent to i by processors that i has chosen to listen to in phase 1.

Note that, in phase 2, we are assuming a *fully connected, point-to-point* communication model, with no restriction on the length of messages. Note also that we could allow processors to choose new random bits at each round without changing the power of our model; we have made the randomisation part of the initialisation only to simplify the description of the actions in a round.

A *protocol* is determined by the function r of all of the processors, along with a bound $f(n)$ on the number of rounds. At the end of $f(n)$ rounds, each processor outputs a bit.

The *adversary* determines the initial assignment and must choose t processors to *corrupt* before the start of the protocol. During the execution of the protocol, the adversary controls all messages sent by the corrupt processors and has access to all messages received by them. We assume private communication links between uncorrupt processors, so that the adversary does not have access to the messages between uncorrupt processors. Moreover, the adversary does not have access to the random bits of uncorrupt processors. A corrupt processor is *unlimited* in the number of messages it may send or receive in a given round.

Every bit assignment, adversary and selection of random bits determine a *run* of a

protocol. The *output* of a run of a protocol P is the bit that the majority of uncorrupt processors output. We say a protocol run has *loss* l if l uncorrupt processors output bits which differ from the output of the run. For every protocol we will say the protocol accepts loss of l, meaning that executions will be regarded as *successful* if the majority bit was held by at least one uncorrupt processor, and the loss of the execution was at most l. We will also consider protocol runs with *no adversary*. In this case an initial bit assignment B is given, and no processors are corrupt.

We call this model of distributed computation the *scalable synchronous distributed network* model.

We will consider protocols that compute *Byzantine agreement*. The goal of such a protocol is to output a bit b, where b is the initially assigned bit of at least one of the uncorrupt processors.

Chapter 4

Adversary Description

Theorem 1. *Any randomised protocol that, with probability* $> \frac{1}{2}+1/\log n$*, where* n *is the number of processors, computes Byzantine Agreement in the scalable synchronous distributed network model containing* n *processors of which* t *are corrupt and that runs for no more than* f *rounds has a loss greater than* $l = \frac{t^{2/3}}{16fn^{1/3}\log^{5/3} n}$*. In particular, if* t *is a constant fraction of processors, then* $\tilde{\Omega}(n^{1/3})$ *rounds are required to achieve agreement among all the uncorrupt processors.*

To compare this result with the upper bound of $O(n/\log n)$ in [11], if we set t = a constant fraction of n, and $f = \log n/\log\log n$, we get that the loss must be $\Omega(n^{1/3}\log\log n/\log^{8/3} n)$.

We will prove this theorem by constructing an adversary, and showing that any protocol must either have at best a $\frac{1}{2}+1/\log n$ probability of success, or must have

loss greater than l in some cases.

We use the following notation: We denote the set of processors $N = \{1, 2, ..., n\}$. For any set of processors, $\bar{A}$ denotes $N - A$. We use $x \leftarrow y$ to indicate that processor x listens for a message from y at some point, and $x \rightarrow y$ to indicate that processor x sends a message to y at some point. We use a superscript after the arrow to restrict this communication. For example $x \rightarrow^i y$ means that processor x sends processor y a message in the i^{th} round.

Our adversary will work by attempting to isolate a set of processors, which we will call the *target set*, from the rest of the processors. These isolated processors will then be given randomly chosen messages according to a distribution different from the distribution that would be sent if the adversary took no action, so that with some probability, the majority of processors in this set will output a bit that differs from the majority of the remainder. This will translate into loss of at least one half of the target set.

We will refer to something called the *message distribution* of a particular input. This refers to the probability distrubution of messages sent and their contents. The probability of a given message being sent depends on the bit assignment given to the system, the sending processor's random bits, and the messages it receives. As such, the message distribution in a given round depends on the distribution in the previous rounds. Unless otherwise noted, when refering to the message distribution we assume

that the adversary is taking no actions, and so the messages will only be determined by the processors' random strings, the input, and the protocol.

To isolate a set of processors, we must have the adversary corrupt all processors that are likely to send to a processor in the set, or that are likely to listen to a processor in the set. Here, likely means with probability at least p, where p will be some value in the interval $(0, 1)$. The value we will use for p will be the largest one possible, giving the largest set of isolated nodes possible, while still remaining small enough to allow us to prove theorem 1. For a processor x, and conditioned on initial assignment B, we will define two *friend* sets, $S_B(x)$ and $L_B(x)$, which are the set of processors likely to send to x and listen to x respectively. To isolate as large a target set as possible, we want to target the processors that have the smallest friend sets. The following Lemma gives a bound on the number of processors x with small friend sets, where small means at most some value u, which we will define in terms of the parameter p.

Lemma 1. *Assume a given protocol and assume that the corrupt processors follow the protocol. For any processor x, any initial bit assignment B, any $u \geq 1$ and any probability $0 < p < 1$*

1. *Let $S_B(x) = \{y | Pr(y \rightarrow x) > p\}$. Let $S_B = \{x \mid |S_B(x)| \leq u\}$. Then $|\overline{S_B}|up \leq fn \lg n$;*

2. *Let* $L_B(x) = \{y | Pr(y \leftarrow x) > p\}$. *Let* $L_B = \{x \mid |L_B(x)| \leq u\}$. *Then* $|\overline{L_B}|up \leq fn \lg n$;

Proof. We prove Statement (1). Due to the identical limits placed on the number of messages sent and the number of processors listened to, the proof of Statement (2) will be identical. Let $X_{i,j}$ denote the indicator variable which is 1 iff processor i sends to processor j, and 0 otherwise. Then the total number of messages sent $X \geq \sum_{i,j} X_{i,j}$ and $E[X] \geq \sum_{i,j} E[X_{i,j}] \geq \sum_{i \in N, j \in \overline{S_B}} E[X_{i,j}] \geq \sum_{i \in \overline{S_B}} |S_B(i)|p \geq |\overline{S_B}|up$. Since there are n processors, each processor can only send $\lg n$ messages per round, and there are only $f(n)$ rounds, we are limited to $n \lg n f(n)$ messages total for the entire protocol. The result follows by observing that the $|\overline{S_B}|up$ is by definition a lower bound on the expected number of messages sent in the system, and that the expectation of the number of messages is bounded above by the maximum number allowed. □

At this point it should be noted that we will be assuming that our value of n is sufficiently large, so that our expressions for the sizes of various sets are integer values, and greater than one. This lemma shows that, given any bit assignment, there is a lower bound on the number of processors such that at most u processors have a probability at least p of sending or listening to them. As such, we can isolate this set of processors, or a subset of it, by, for each processor in the set, taking over the at most $2u$ processors in $S_B(x)$ and $L_B(x)$. The size of the subset, and the values u

and p will have to be chosen such that the total number of processors to be corrupt is at most t, the number of processors our adversary can corrupt. The initial target set will be the intersection of S_B and L_B, and we will refer to it as T.

Once these processors are isolated, we must give them a different distribution of messages. We want to give them a distribution of messages as though the system were given some other bit assignment B', so that the target set will have a different output. This B' must have the property that $B(x) = B'(x)$ for all x in the target set. This is because once x has been given a bit, we cannot then attempt to send x a distribution of messages consistent with a bit assignment in which it received a different bit, since depending on the protocol, x could send significantly different messages due to the different initial bit.

It may be the case that a processor in the target set is not likely to have some other processor y send to it given B, but is very likely to receive a message given B'. Thus, to properly convince our isolated processors, we must also have the adversary take over $S_{B'}(x)$ and $L_{B'}(x)$. However, Lemma 1 only applies to a single bit assignment, so these sets could be much larger than u. Therefore, we must show that there are two bit assignments B, B' that will produce a different output with over $1/2 + 1/\log n$ probability, such that the intersection of $S_B, L_B, S_{B'}$ and $L_{B'}$ will be as large as possible, to force as large a loss as possible.

In Lemma 2 we will state this requirement more formally, and give a proof that

such bit assignments exist. The proof of the lemma will make use of the following observation about bit assignments where almost all of the processors have the same initial bit.

Observation 1. *Let P be any protocol which computes Byzantine Agreement with probability $p > \frac{1}{2}$ and with loss of less than l for adversaries which corrupt at most $t < \frac{n}{2}$ processors. Then when P is run without an adversary on any initial assignment where at least $n - t$ bits have the value b, with probability p the output of P must be b with loss less than l.*

Proof. Consider an adversary which corrupts the t processors which hold bits different from b, but these processors otherwise act like uncorrupt processors. By assumption, P must output b with loss less than l with probability p. On the other hand, P can't distinguish this case from the case where fewer or different processors are corrupt. □

In the following Lemma, we will attempt to establish a set processors to corrupt, which we will call C, such that $|C| \leq t$, and a target set T' with $|T'| = t/4u$. As mentioned above, the second bit assignment depends on our choice of the first, so they must be chosen in order. We will call the target set for just the first bit assignment T. The T' we choose must be a subset of this T. C will be the union of the four friend sets $S_B(x), L_B(x), S_{B'}(x)$ and $L_{B'}(x)$ for every x in the target set T'. T' will be the intersection of $S_B, S_{B'}, L_B$, and $L_{B'}$, the four sets of processors with small friend sets.

Because it will be acceptable for two processors in the target set to communicate with each other without breaking the isolation of the target set, we will remove from C any processors that are in T'. To get our result we will fix the product of size u and probability p, where u and p are the parameters defined in Lemma 1, and leave only parameter p to be decided upon by the adversary.

Lemma 2. *Let P be any protocol which computes Byzantine Agreement with probability $> \frac{1}{2}$ for adversaries which corrupt at most t processors. Suppose that we have values u, p such that $up \geq \frac{8f(n)n \lg n}{t}$. Then there are two bit assignments B and B' and three sets of processors C, T, and T' such that*

1. *When run without an adversary, the output of P on B is 0 with probability $> \frac{1}{2}$ and the output of P on B' is 1 with probability $> \frac{1}{2}$.*

2. *$T' \subset T - C, |T'| = t/(4u)$ and $|C| \leq t$; For brevity we will set $g = t/(4u)$ to represent the size of this set.*

3. *For all processors $x \in T$, $B(x) = B'(x) = 0$ and for all $x \in \overline{T}$, $B(x) = 0$ and $B'(x) = 1$.*

4. *For all processors x, y such that $y \in \overline{C}$, $x \in T'$, $y \notin \overline{T'} \cap (S_B(x) \cup L_B(x) \cup S_{B'}(x) \cup L_{B'}(x))$*

Proof. We construct the assignments B and B'. Let B be the all zeros assignment. For any protocol P, it follows by Lemma 1 that for the given choice of u and p,

$|\overline{S_B}| \leq \frac{t}{8}$ and $|\overline{L_B}| \leq \frac{t}{8}$. But then

$$|S_B \cap L_B| = n - |\overline{S_B} \cup \overline{L_B}| \geq n - 2\frac{t}{8} \geq \frac{3n}{4},$$

so we set $T = S_B \cap L_B$. As t must always be at most n, $|T| > \frac{t}{2}$. Now let B' be the assignment in which all processors in T have 0 and all other processors have 1. Again by Lemma 1 we have $|\overline{S_{B'}}| \leq \frac{t}{8}$ and $|\overline{L_{B'}}| \leq \frac{t}{8}$. But then $|\overline{S_{B'}} \cup \overline{L_{B'}}| \leq \frac{t}{4}$, so it follows that

$$|T \cap S_{B'} \cap L_{B'}| = |T - (\overline{S_{B'}} \cup \overline{L_{B'}})| \geq \frac{t}{4}$$

Then, since $u \geq 1$, there is a set $T' \subset T \cap S_{B'} \cap L_{B'}$ of size $\frac{t}{4u}$. We chose our T' in an arbitrary way, as any subset of this size will be sufficient for our purposes.

Define

$$C = \bigcup_{x \in T'} (\overline{T'} \cap (S_B(x) \cup L_B(x) \cup S_{B'}(x) \cup L_{B'}(x)))$$

Statement (1) follows from Observation 1. For statement (2)

$$\begin{aligned} |C| &\leq + \sum_{x \in T'} (|S_B(x)| + |L_B(x)| + |S_{B'}(x)| + |L_{B'}|) \\ &\leq (4u)\frac{t}{4u} \\ &= t \end{aligned}$$

Statement (3) follows from the definition of B and B'. Statement (4) follows from the construction of C. If there is an $x \in T'$ such that $y \in S_B(x) \cup L_B(x) \cup S_{B'}(x) \cup L_{B'}(x)$ then either $y \in T'$ or $y \in C$. □

4.1 The adversary strategy

Given a protocol P, the adversary determines two initial assignments B and B' with properties which satisfy Lemma 2 and the sets $S_B, L_B, T, S_{B'}, L_{B'}$ and the set C which it corrupts.

We will consider two runs of the system, with two different adversary strategies. Under the first strategy, the processors are given bit assignment B', and adversary takes no action. That is, while it corrupts the set C, the corrupt processors will behave exactly as they would were they not corrupt.

Under the second strategy, the processors are given bit assignment B, and the adversary will corrupt the set C. Below is the description of how the adversary will determine what action these corrupt processors take.

Concurrently with the execution of the protocol, the adversary runs two simulations sim_B and $sim_{B'}$ of P, with respective initial assignments B and B'.

Each simulation will contain all of the processors in the system. For each simulation, there will be two different kinds of processors. A simulation will contain *simulated processors*, and it will contain *stub processors*. A simulated processor has its behaviours simulated by the adversary. The adversary simulates its state, and has it behave according to this simulated state using the protocol P. A stub processor's behaviour is not simulated, but is based on the behaviour of the corresponding real processor. If a processor x exists as a stub processor in sim_B (or $sim_{B'}$), then for

every message a corrupt processor receives from x, the corresponding stub of x in sim_B ($sim_{B'}$) will send the same message to the simulated version of that corrupt processor.

We will now describe the sets of simulated and stub processors for our two simulated systems.

In $simB$, each simulated processor has its initial bit set to 0. The processors are divided as follows:

- The set of simulated processors will be $C \cup T'$.
- The set of stub processors will be the remainder.

In $sim_{B'}$,the simulated processors in $\overline{T'}$ have initial bits set to 1; the simulated processors in T' have initial bits set to 0. The processors are divided as follows:

- The set of simulated processors will be $\overline{T'}$
- The set of stub processors will again be the remainder, T'

In the real execution, the adversary will give all processors initial bit 0. During the execution of round k of the protocol, for all $x \in C$, the messages sent by x will correspond to the messages sent by x in both sim_B and $sim_{B'}$, as follows.

- if $y \in \overline{T}$ and $x \rightarrow^k y$ in sim_B then x sends the message as in sim_B.
- if $y \in T'$ and $x \rightarrow^k y$ in $sim_{B'}$, then x sends the message as in $sim_{B'}$

Note that this strategy will result in each corrupt processor sending up to $2 \log n$ messages per round. We will now prove that for this adversary strategy, with high probability the uncorrupt processors in T' are isolated from $\overline{T'}$. A processor x is *isolated* from a set S of processors, if for every uncorrupt processor $y \in S$ and all k, if $y \rightarrow^k x$ then it is not the case that $x \leftarrow^k y$, and if $x \rightarrow^k y$ then it is not the case that $y \leftarrow^k x$. We first show that with high probability a single processor is isolated.

Lemma 3. *Let g, p and T be defined as in the previous lemmas. Then in any protocol which runs for no more than f rounds, the following statements hold with probability $1 - 8fg \log n \sqrt{p}$*

1. *Each processor $x \in T'$ has been isolated in all rounds for the adversary strategies and both initial assignments described above.*

2. *For the first adversary strategy, using initial assignment B', the distribution of messages received by all uncorrupt processors is identical to that received when there are no corrupt processors.*

3. *For the second adversary strategy, using initial assignment B, the distribution of messages received by all uncorrupt processors in T is identical to the distribution of messages under P when run on initial assignment B with no corrupt processors and the distribution of messages received by all uncorrupt processors in T is identical to the distribution of messages under P when run on initial*

assignment B' with no uncorrupt processors.

Proof. The proof is by induction on the number of rounds f.

Base Case: Initially, every processors is isolated from every other processor, and no messages have been received.

Induction Step: Assume that in round i, the processors in T remain isolated and the distribution of messages is consistent with the Induction Statement. We will use the sets $S_B, L_B, S_{B'}$, and $L_{B'}$ as described in Lemma 1. We now prove item (1) of the Induction Statement for round $i+1$. In the proof below, we assume x is any uncorrupt processor in T. Let us describe some events for this fixed x and for round i.

- F_i is the event that x remains isolated for the first i rounds.
- P_i is the event that $(y \rightarrow^i x, x \leftarrow^i y)$ for some uncorrupt processor y.
- Q_i is the event that $(x \rightarrow^i y, y \leftarrow^i x)$ for some uncorrupt processor y.
- $Lis_i(Y)$ is the event that, in round i, x listens to exactly the set of uncorrupt processors Y. It could listen to additional corrupt processors, but it listens to all processors in Y, and no other uncorrupt processors.
- $Send_i(Y)$ is the event that, in round i, x sends to exactly the set of uncorrupt processors Y. Again, it is possible that it also sends to some corrupt processors.

Each of these events will be viewed as a set of complete communications for the first i rounds. A complete communication is every message sent by every processor, and every processor that every other processor listened to. A complete communication fully describes all information shared between the processors. The event that a set of communications $\{c_1, c_2 \ldots\}$ occurs is equal to the event that c_1 occurs, or the event that c_2 occurs, and so forth. Note that since these events describe the set of complete communications these are elementary events, i.e., $Pr(c_i \wedge c_j) = 0$.

Definitions: We define subsets of F_i:

$$\begin{aligned} \alpha_{i,y} &= \{c \in F_i \mid Pr(y \rightarrow^{i+1} x \mid c) > \sqrt{p})\} \\ \beta_{i,y} &= \{c \in F_i \mid Pr(y \leftarrow^{i+1} x \mid c) > \sqrt{p})\} \end{aligned}$$

For a set of processors Y, we define

$$\begin{aligned} \alpha_{i,Y} &= \bigcup_{y \in Y} \alpha_{i,y} \\ \beta_{i,Y} &= \bigcup_{y \in Y} \beta_{i,y}. \end{aligned}$$

The complement of these sets will be defined in the context of F_i, e.g. $\overline{\alpha}_{i,Y} = F_i \setminus \alpha_{i,Y}$.

Claim 1. *: For any $y \in \bar{T} - C$, $Pr(\alpha_{i,y}) \leq \sqrt{p}$ and $Pr(\beta_{i,y}) \leq \sqrt{p}$.*

Proof of Claim: By Lemma 2 for any uncorrupt processor $y \in \overline{T}$,

$$\begin{aligned} p &\geq Pr(y \rightarrow x) \\ &\geq Pr(y \rightarrow x \wedge \alpha_{i,y}) \\ &= Pr(y \rightarrow x | \alpha_{i,y}) Pr(\alpha_{i,y}) \\ &\geq \sqrt{p} Pr(\alpha_{i,y}) \end{aligned}$$

The last line follows by the definition of $\alpha_{i,y}$. Hence $\sqrt{p} \geq Pr(\alpha_{i,y})$. A similar argument shows $Pr(\beta_{i,y}) \leq \sqrt{p}$.

Claim 2. *: $Pr(F_i \wedge P_{i+1}) \leq 2 \log n \sqrt{p}$, and $Pr(F_i \wedge Q_{i+1}) \leq 2 \log n \sqrt{p}$.*

It follows from Claim 2 that the probability of failing to isolate one processor in T' in the $(i+1)^{th}$ round is less than $4 \log n \sqrt{p}$. Hence, taking union bounds, the probability that any of the g processors are not isolated in the $(i+1)^{th}$ round is less than $4g \log n \sqrt{p}$. Note that in the case of the second adversary strategy, we want the simulated set T' to also be isolated. So, for the simulated system, the same bound will apply, and we will take the union bound of the probabilities that either system has a set T' that is not isolated, getting a total probability of $8g \log n \sqrt{p}$. As such, the probability that any of the g processors are not isolated in any rounds up to and including round $i+1$, and in either system in the case of the second strategy, is less than $8g \log n \sqrt{p} + 8gi \log n \sqrt{p}$, which establishes item (1) of the induction.

It remains to prove Claim 2.

Proof of Claim: Let G denote the uncorrupt processors. Then

$$\Pr(P_i) \leq \sum_{Y \subset G} \sum_{y \in Y} \Pr(y \rightarrow^i x \wedge Lis_i(Y))$$

Observe that $\Pr(F_i \wedge P_{i+1}) = \sum_{c \in F_i} Pr(c \wedge P_{i+1})$, where c is some complete communication over the first i rounds. By our definitions, this is bounded above by

$$\sum_{c \in B_i} \sum_{Y \subset G} \sum_{y \in Y} Pr(c \wedge Lis_{i+1}(Y) \wedge y \rightarrow^{i+1} x)$$

We can rearrange the sums to have $\sum_{Y \subset G} \sum_{c \in B_i} \sum_{y \in Y}$. Further, we will split the sum of communications into two separate sums. Our probability is now

$$\begin{aligned} Pr(B_i \wedge P_{i+1}) &= \sum_{Y \subset G} \sum_{c \in \alpha_{i,Y}} \sum_{y \in Y} Pr(c \wedge Lis_{i+1}(Y) \wedge y \rightarrow^{i+1} x) \qquad (4.1) \\ &+ \sum_{Y \subset G} \sum_{c \in \bar{\alpha}_{i,Y}} \sum_{y \in Y} Pr(c \wedge Lis_{i+1}(Y) \wedge y \rightarrow^{i+1} x) \qquad (4.2) \end{aligned}$$

We will now bound terms (4.1) and (4.2). For term (4.1), we have

$$\begin{aligned} \sum_{Y \subset G} \sum_{c_i \in \alpha_{i,Y}} \sum_{y \in Y} Pr(c \wedge Lis_{i+1}(Y) \wedge y \rightarrow^{i+1} x) &\leq \sum_{Y \subset G} \sum_{c \in \alpha_{i,Y}} \sum_{y \in Y} Pr(Lis_{i+1}(Y) \wedge c) \\ \text{(Since } |Y| \leq \log n \text{ if } Pr(Lis_{i+1}(Y) > 0) &\leq \log n \sum_{Y \subset G} \sum_{c_i \in \alpha_{i,Y}} Pr(Lis_{i+1}(Y) \wedge c) \\ &= \log n \sum_{Y \subset G} Pr(Lis_{i+1}(Y) \wedge \alpha_{i,Y}) \\ &= \log n \sum_{Y \subset G} Pr(Lis_{i+1}(Y) | \alpha_{i,Y}) Pr(\alpha_{i,Y}) \\ &\leq \log n \sqrt{p} \sum_{Y \subset G} Pr(Lis_{i+1}(Y) | \alpha_{i,Y}) \\ &\leq \log n \sqrt{p} \end{aligned}$$

To bound term (4.2), we first note that for a fixed communication description c, the actions of any two processors x and y are independent given c since their actions are based only on their private random strings and the messages they received (or didn't). E.g., if A is an action taken by x and B is an action taken by y, $Pr(A|c \wedge B) = Pr(A|c)$

Given the above observation, we have, using Bayes' rule

$$\begin{aligned} Pr(c \wedge Lis_{i+1}(Y) \wedge y \rightarrow^{i+1} x) &= Pr(y \rightarrow^{i+1} x | c \wedge Lis_{i+1}(Y)) Pr(c \wedge Lis_{i+1}(Y)) \\ &= Pr(y \rightarrow^{i+1} x | c) Pr(c \wedge Lis_{i+1}(Y)) \end{aligned}$$

This allows us to bound term (4.2) by

$$\begin{aligned} \sum_{Y \subset G} \sum_{c \in \bar{\alpha}_{i,Y}} \sum_{y \in Y} & Pr(y \rightarrow^{i+1} x | c) Pr(c \wedge Lis_{i+1}(Y)) \\ &\leq \sum_{Y \subset G} \sum_{c \in \bar{\alpha}_{i,Y}} \sum_{y \in Y} \sqrt{p} Pr(Lis_{i+1}(Y)) \\ &= \log n \sqrt{p} \sum_{Y \subset G} \sum_{c \in \bar{\alpha}_{i,Y}} Pr(Lis_{i+1}(Y) \wedge c) \\ &= \log n \sqrt{p} \sum_{Y \subset G} Pr(Lis_{i+1}(Y) \wedge \bar{\alpha}_{i,Y}) \\ &\leq \log n \sqrt{p} \end{aligned}$$

Thus, our total probability will be at most $2 \log n \sqrt{p}$. An identical argument, using

Q_{i+1}, and our β sets, will give us that $Pr(F_i \wedge Q_{i+1}) \leq 2 \log n \sqrt{p}$ as well, which ends the proof of Claim 2.

To prove item (2) we note that the corrupt processors act as uncorrupt processors so the distribution of messages should be the same as if there are no corrupt processors.

To prove item(3): By induction, the processors in T' have received messages in rounds 1 through i in a distribution identical to what would be sent under P with no corrupt processors under assignment B'. In the $(i+1)^{th}$ round they are isolated, hence they receive only messages from corrupt processors. The adversary determines which messages to send according to $sim_{B'}$, in which the corrupt processors act like uncorrupt processors given the actual messages from processors in T'. Hence the distribution of messages received by processors in T' in the $(i+1)^{th}$ round is indistinguishable from what they would have received under P with initial assignment B' and with no corrupt processors. Similarly, since by the induction hypothesis the uncorrupt processors in $\overline{T'}$ are isolated from T', and have received messages consistent with B and no adversary, and have received messages from C according to Sim_B, the distribution of messages will likewise remain indistinguishable from what they would have received under P with initial assignment B, concluding the proof of the induction statement.

The proof of the lemma follows from this induction statement. □

4.2 Proof of Theorem

Using the above lemmas and observations, we can now prove Theorem 1.

Proof of Theorem 1. Let P be any protocol that computes Byzantine Agreement with probability at least $1/2+\frac{1}{\log n}$, and let B and B' be the assignments defined in Lemma 2 for this protocol. We will consider two adversary strategies. Under the first strategy, the system will be given bit assignment B', and the adversary will not corrupt any processors. Under the second strategy, the system will be given bit assignment B and the adversary will behave as described in Section 4.1.

We have shown that with probability at least $1-8fg\sqrt{p}\log n$, the g processors in the set T' are isolated from the uncorrupt processors outside of T', when P is run on either B' using the first adversary strategy, or on B using the second. Let us assume that this event occurs, and T' is isolated from the rest of the uncorrupt processors. We must consider the possible loss.

Let us assume that when running P on B' given that T' remains isolated, there is at least probability $1/2$ of at least half of T' outputting 1. Now, consider the result of running P on B using the second adversary strategy. By Lemma 3, the distribution of messages received by T', given that T' remains isolated, is the same as it would have received on B' with the first strategy. Since P has at least probability $1/2$ of having at least half of T' output a 1 when run on B', the same probability will apply

to at least half of T' outputting a 1 when P is run on B under the second strategy, also resulting in probability at least $1/2$ of having loss at least $g/2$. (This is because by definition the majority must be 0 on B, so any 1 output is a loss)

Now, given that T' is isolated, there is at least a $1/2$ probability that the loss is at least $g/2$. So, with probability at least $(1/2)(1 - 8fg \log n\sqrt{p})$, P will have loss at least $g/2$.

The loss can be less than $g/2$ if either this event does not occur, or T' was not isolated after f rounds. The probability of either event occuring is at most the sum of these two probabilities. Adding the two probabilities gives us that the loss will be at most $g/2$ with probability at most $1/2 + 4fg \log n\sqrt{p}$.

To contradict our protocol's definition, we want this probability to be $1/2 + 1/\log n$, so we set $4fg\sqrt{p}\log n = 1/\log n$, and get that we should use the value $p = (16f^2g^2 \log^4 n)^{-1}$. Using this probability value in our previous definitions of g and u gives us that $g = \frac{(t^2)^{1/3}}{8fn^{1/3}\log^{5/3} n}$, and therefore, $l = \frac{(t^2)^{1/3}}{16fn^{1/3}\log^{5/3} n}$. In other words, if we use this value of p for our adversary, then any protocol that has probability of success greater than $1/2+1/\log n$ and tolerates loss of at most $l = \frac{(t^2)^{1/3}}{16fn^{1/3}\log^{5/3} n}$ results in a contradiction, and so such a protocol must accept loss greater than this value if it is to compute Byzantine agreement with probability greater than $1/2 + 1/\log n$.

Now, let us assume the converse, that when running P on B' given that T remains isolated, there is greater than probability $1/2$ of at least half of T outputting a 0.

This means that either there is loss at least $g/2$, or the majority was 0. By the same argument above, and the same choice for p, the probability will be at least $1/2 - 1/\log n$ of either having a 0 majority, or having loss at least $g/2$. This precludes the possibility of P outputting 1 with loss less than $g/2$ with probability greater than $1/2 + 1/\log n$, and so by Observation 1, we again get that P must tolerate a loss of more than $l = \frac{(t^2)^{1/3}}{16fn^{1/3}\log^{5/3} n}$ if it is to compute Byzantine agreement with probability greater than $1/2 + 1/\log n$.

Thus, for either case for the majority output of T, P must tolerate loss greater than $l = \frac{(t^2)^{1/3}}{16fn^{1/3}\log^{5/3} n}$ if P is to output a valid majority with probability greater than $1/2 + 1/\log n$, as desired. □

Chapter 5

Conclusion and open problems

Our initial goal for this paper was to address the open problem in [11], namely, in a point-to- point full information network model where each uncorrupt processor is restricted to sending polylogarithmic number of bits over the course of the protocol, could Byzantine agreement be computed with no loss? Here we are able to prove a lower bound which shows a tradeoff between the number of rounds and the loss, and demonstrates that if the number of corrupt processors is a constant fraction of the total then $\Omega(n^{1/3})$ rounds are needed to achieve no loss. We have loss of $\Omega(n^{1/3} \log\log n / \log^{8/3} n)$ when using parameters similar to those used in [11], in contrast with their $O(n/\log n)$ upper bound. While we have required in our model no more than $\log n$ messages be sent by each processor per round, it is not hard to see that a similar result would follow, with an extra polylogarithmic divisor, if we allowed

polylogarithmic messages per round instead. We observe that our proof goes through with an initial assignment in which all but $O(n^{1/3})$ of the uncorrupt processors agree on the same bit. In some sense we were more successful then we had hoped: our bounds hold even if we give the network extra power: messages can be of arbitrary length, communications between uncorrupt processors can be private from the adversary, and the protocol need only succeed with probability only slightly greater than 1/2. The protocols in [11] were designed to prevent the adversary from "flooding" a processor by assuring that at each step each processor knows the set of uncorrupt processors who might send to it. Our lower bound applies to protocols of this type. But our lower bound (and the upper bound in [11]) do not address what would happen if we allowed uncorrupt processors to listen for any messages received and select out of those a set of log n to process based on some function of the senders' address, or even if we allowed them to process all the messages they received, and restrict only the number of messages sent. This question remains tantalisingly open. Both [11] and this thesis assume a non-adaptive adversary, which is the only kind reasonable to consider in a leader election protocol; the problem of Byzantine Agreement in a scalable network with a malicious adaptive adversary is yet to be considered. Both works assume that corrupt processors may send and receive an arbitrary number of messages. In fact, the corrupt processors send only as many as twice the number that uncorrupt processors are allowed to send. If we limit the corrupt processors to

the same number of messages as the uncorrupt processors then it is possible that Byzantine Agreement with reduced or no loss may be possible in a scalable network. Computing Byzantine Agreement in a model with a bounded degree network with polylogarithmic number of messages is still open.

Bibliography

[1] Bar-Joseph, Z., and Ben-Or, M. A tight lower bound for randomized synchronous consensus. In *PODC '98: Proceedings of the seventeenth annual ACM symposium on Principles of distributed computing* (New York, NY, USA, 1998), ACM Press, pp. 193–199.

[2] Christin, N., Weigend, A. S., and Chuang, J. Content availability, pollution and poisoning in file sharing peer-to-peer networks. In *EC '05: Proceedings of the 6th ACM conference on Electronic commerce* (New York, NY, USA, 2005), ACM Press, pp. 68–77.

[3] Cloudmark website. http://cloudmark.com/.

[4] Dwork, C., Peleg, D., Pippenger, N., and Upfal, E. Fault tolerance in networks of bounded degree. In *STOC '86: Proceedings of the eighteenth annual ACM symposium on Theory of computing* (New York, NY, USA, 1986), ACM Press, pp. 370–379.

[5] Feige, U. Noncryptographic selection protocols. In *FOCS '99: Proceedings of the 40th Annual Symposium on Foundations of Computer Science* (Washington, DC, USA, 1999), IEEE Computer Society, p. 142.

[6] Feldman, P., and Micali, S. An optimal probabilistic protocol for synchronous byzantine agreement. *SIAM J. Comput. 26*, 4 (1997), 873–933.

[7] Fich, F., and Ruppert, E. Hundreds of impossibility results for distributed computing. *Distrib. Comput. 16*, 2-3 (2003), 121–163.

[8] Fischer, M. J., and Lynch, N. A. A lower bound for the time to assure interactive consistency. *Information Processing Letters 14*, 4 (1982), 183–186.

[9] Folding@home website. http://www.stanford.edu/group/pandegroup/folding/.

[10] Gnutella website. http://www.gnutella.com/.

[11] King, V., Saia, J., Sanwalani, V., and Vee, E. Scalable leader election. In *SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm* (New York, NY, USA, 2006), ACM Press, pp. 990–999.

[12] Lamport, L., Shostak, R., and Pease, M. The byzantine generals problem. In *Advances in Ultra-Dependable Distributed Systems, N. Suri, C. J. Walter, and M. M. Hugue (Eds.), IEEE Computer Society Press.* 1995.

[13] Lewis, C., and Saia, J. Scalable Byzantine agreement. Tech. rep., University of New Mexico, 2004.

[14] Russell, A., Saks, M., and Zuckerman, D. Lower bounds for leader election and collective coin-flipping in the perfect information model. pp. 339–347.

[15] Russell, A., and Zuckerman, D. Perfect information leader election in $log^* n + o(1)$ rounds. In *FOCS '98: Proceedings of the 39th Annual Symposium on Foundations of Computer Science* (Washington, DC, USA, 1998), IEEE Computer Society, p. 576.

[16] Seti@home website. http://setiathome.berkeley.edu/.