Agreement In the Presence of Faults, On Networks of Bounded Degree

Michael Ben-Or
Institute of Computer Science
Hebrew University
Jerusalem, Israel

Dana Ron*
Laboratory of Computer Science
MIT
Cambridge, MA, USA

Abstract

We present networks of bounded degree and a fully polynomial almost everywhere agreement scheme which tolerate, with high probability, randomly located faulty processors, where processors fail independently with some constant probability.

1 Introduction

Dwork et. al. [DPPU] introduced the notion of *almost everywhere agreement*. They exhibited a communication network of bounded degree connecting N processors along with a polynomial protocol enabling all but $O(t)$ of the correct processors to reach agreement in the presence of up to $t = O(N/\log N)$ faulty processors. Further work by Berman and Garay [BG1, BG2] improved the efficiency of these protocols.

Handling more faults is a considerably more difficult problem. Note that in networks of bounded degree the distance between most pairs of nodes is $\Omega(\log N)$. Therefore, even if the faulty processors are picked randomly, each independently with probability $\epsilon \gg 1/\log N$, then with high probability most communication paths between most pairs of correct processors will contain a faulty processor.

Recently Upfal [U] proved that for any strong enough expander there is an almost everywhere agreement protocol tolerating linearly many faulty processors. The communication complexity of this protocol is polynomial but the local computation required by the processors themselves is exponential. Thus the problem of constructing a network of bounded degree with a polynomial, almost everywhere, agreement protocol remains open.

In this work we treat the case in which the faulty processors are selected randomly. We construct a bounded degree network and a polynomial agreement protocol such that the following holds. If each processor fails independently with probability ϵ, $0 < \epsilon < \epsilon_c$, where ϵ_c is some constant smaller than 1, then with high probability the network will reach almost everywhere agreement.

*This research was done while the second author was at the Hebrew University.

2 Definitions, Notations and Results

Our model of communication is a synchronous (possibly sparse) network of N processors. A protocol specifies a set of programs for the participating processors. These programs are partitioned into rounds which can include: sending a message to each neighbouring processor, receiving such messages and doing some local computations. A correct processor acts exactly according to the protocol. Faulty processors may coordinate their actions in the worst "Byzantine" way. We shall refer to the following version of Byzantine agreement: Each processor begins with an initial binary value. By the end of the agreement protocol, all the correct processors must agree on a common value. If all the correct processors hold the same initial value, then this must be the agreement value.

Since Byzantine agreement is achievable only if the number of faulty processors is less then one-half of the connectivity of the network [D], Dwork et. al. [DPPU] suggest a relaxation of the standard definition of Byzantine agreement. Instead of requiring that *all* correct processors must reach a common decision, they require that all *but a small fraction* of the correct processors reach a common decision. Namely,

Definition 2.1 [DPPU] *Let G be a network, consisting of N processors, and T and X be integers so that $X < \frac{N-T}{2}$. A protocol P achieves* **T-resilient X-agreement** *on G if in every execution of P on G, in which at most T processors fail, all but X of the correct processors decide on a common value. Furthermore, if all correct processors hold the same initial value then that must be the value chosen. The (up to X) correct processors which do not arrive at the majority decision are called* **excluded** *processors.*
Let t and x be integer functions, and $\mathcal{G} = \{G_N\}_{N \in \mathcal{N}}$ be a family of networks so that G_N consists of N processors. A protocol P achieves **$t(\cdot)$-resilient $x(\cdot)$-agreement** *on the family $\mathcal{G}$ if for all sufficiently large $N \in \mathcal{N}$, the protocol P achieves $t(N)$-resilient $x(N)$-agreement on G_N.*

Based on this definition we quote the following result.

Theorem 2.1 [DPPU] *There exist a constant $0 < \alpha < 1$, an explicitly constructible family of networks $\{G_N\}$ of constant degree, and an agreement protocol P, such that P admits a $t(N)$-resilient $(t(N)+1)$-agreement on $\{G_N\}$, for $t(N) \leq \alpha \frac{N}{\log N}$.*
Furthermore, if a majority of $\frac{2}{3}N$ of the correct processors hold the same initial value, then that will be the agreed value.

Before we state our main result, we further relax the requirement of Definition 2.1 as follows. Instead of requiring that $x(\cdot)$-agreement be reached for any choice of $t(\cdot)$ faulty processors, we only require that $x(\cdot)$-agreement be reached for an overwhelming majority of the choices of $t(\cdot)$ faulty processors. Namely,

Definition 2.2 *Let $0 \leq \epsilon, \delta < 1$ be two constants, G be a network, and X be an integer. A protocol P achieves* **(ϵ, δ)-resilient X-agreement** *on G if, with probability at least $1-\delta$, an execution of P, in which each processor fails independently of the others with probability ϵ, terminates with all but X of the correct processors agreeing on a common value. If all correct processors hold the same initial value then that must be the value chosen.*
A protocol P achieves **$(\epsilon(\cdot), \delta(\cdot))$-resilient $x(\cdot)$-agreement** *on the family $\{G_N\}$ if for all sufficiently large N's, the protocol P achieves $(\epsilon(N), \delta(N))$-resilient $x(N)$-agreement on G_N.*

Based on the above definition we state our result:

Theorem 2.2 *There exist constants $0 < \alpha, \Upsilon < 1$, an explicitly constructable family of networks $\{G_N\}$ of a constant degree, and an agreement protocol P, such that for every $\epsilon \le \Upsilon$, P admits an $(\epsilon, 2^{-\alpha \frac{N}{(\log N)^2}})$-resilient $O(\epsilon N)$-agreement on $\{G_N\}$,*

The main contribution of this result, is in the case where processors fail with a large (constant) probability. We describe this case in detail in the following sections. In the appendix we state the modifications needed for the general case.

3 The Network's Construction

The networks mentioned in Theorem 2.1 are constructed by superimposing special constant degree graphs called *compressor graphs* (see [P],[LPS]) on the *butterfly graph.* We shall refer to these networks as **butterfly-compressor networks** (**b.c.n**'s). The family of networks we construct, uses these butterfly-compressor networks as building blocks. We start with the construction of networks of degree $O(\log^* N)$. In the end of this section we describe shortly how to modify the construction in order to achieve constant degree.

Let $\log^{(k)} N$ be the k'th log of N and $N_i \stackrel{\text{def}}{=} \log^{(i)} N$. m is a constant whose value is set later. s is the largest integer satisfying $N_s = \log^{(s)} N_0 \ge m$ ($s \simeq \log^* N_0$).

The following is the recursive construction of our network:

- Let G_0 be a singleton set consisting of a butterfly-compressor network of N_0 processors and degree d_0.
- For $i = 1$ *to* $s - 1$
  - Replace each processor p in every $g \in G_{i-1}$ by a committee of processors connected as a b.c.n of size $N_i = \log N_{i-1} (= \log^{(i)} N)$ and degree d_0. If p and q were neighbors in g, connect each new processor in the committee replacing p with the corresponding processor in the committee replacing q.

    The resulting set of sub-networks is denoted by G_i. The degree of every $g \in G_i$ is $d_i = d_{i-1} + d_0$.
- Replace each processor p in every $g \in G_{s-1}$ with a fully connected committee of size N_s. If p and q were neighbors in g, connect each new processor in the committee replacing p with every processor in the committee replacing q.

  The set of all these fully connected committees is denoted by G_s.

The constructed network is of size $N = N_0 \cdot (\log N_0) \cdot \ldots \cdot (\log^{(s-1)} N_0) \cdot (\log^{(s)} N_0)$ and degree $s \cdot d_0 + N_s^2$. In order to reduce the degree to a constant, we can add another phase to each iteration. We replace each processor p in the new, stage i network, with a committee of processors connected as a ring of size d_i (the degree of the original processor). Each processor is now connected to its two neighbors on the ring, and to a respective processor in a ring replacing one of the neighbors of p. The degree of each new processor is thus 3. The final network is larger by a factor of $d_0^{\log^* N_0}$, and it's degree is $N_s^2 + 3$.

4 The Agreement Scheme

In the description of the agreement protocol, we use [DPPU]'s agreement protocol on the butterfly-compressor networks as a black box. The top level of our constructed network is a butterfly-compressor network. Each processor was replaced by a committee of processors connected as described in the previous section. Hence, in order to execute [DPPU]'s agreement scheme on our network, we need only describe how the atomic operations of processors in the butterfly-compressor networks are translated into procedures on the committees of processors.

[DPPU] use two atomic operations: sending a message to a neighbouring processor, and receiving a message from a neighbouring processor. Deciding on the majority value among values received can be seen as a special case of receiving. We describe how these operations are implemented on the committees of processors described in the previous section, and how agreement is reached on the networks of degree $O(\log^* N)$. In the end of this section we sketch the modification needed for the constant degree networks.

What follows are the **transmit** and **agree** procedures. The **transmit** procedure implements sending and receiving a message. The **agree** procedure functions as the main procedure when initially called for the complete network, and is also called as a subroutine by **transmit** for the receive phase. On the lowest level committees of processors, which are fully connected, a byzantine agreement, denoted by **BA**, is executed. **BA** can be any byzantine agreement protocol for complete networks, resilient to $t < \frac{1}{3}N$ faulty processors.

Whenever we refer to subcommittees of a level i committee $g \in G_i$, we mean the highest level $(i+1)$ subcommittees of g.

Procedure **transmit**(g_1,g_2)
(Transmitting a message from committee $g_1 \in G_i$ to a neighbouring committee $g_2 \in G_i$)
If $i = s$ then
1. Each processor in g_1 sends the message it holds to every processor in g_2.
2. Each processor in g_2 chooses the majority value among the messages it received.

Else $(i < s)$
Every subcommittee g' in g_1 and it's neighbouring subcommittee g'' in g_2, execute **transmit**(g', g'')
Execute **agree**(g_2).

Procedure **agree**(g)
(an agreement procedure for a committee $g \in G_i$)
If $i = s$ then
g undergoes **B.A.**
Else $(i < s)$
1. Every subcommittee g' in g executes **agree**(g')
2. Run the [DPPU] agreement, exchanging each message passage between processors p_1 and p_2, with a call to **transmit**(g_1, g_2), g_1, g_2 being the respective subcommittees.

When a committee has to decide, according to the [DPPU] protocol, on a majority value among values received from it's neighbors, then the majority decision is made by the individual processors on the lowest level of recursion of the receiving phase, before executing any agreement.

The agreement can be viewed bottom-up instead of top-down. The agreement starts, in practice, with a byzantine agreement in every fully connected committee of processors and proceeds with agreement being executed on the higher level committees, until it reaches the top level of the entire network.

In order to modify the agreement scheme for the constant degree networks, we need only extend the **transmit** and **agree** procedures so that they treat the case of committees connected as a constant size ring. This can be done in a straightforward manner by simply transmitting all values held by the ring members (wether committees or processors), around the ring, and choosing the majority value received.

5 Analysis

In this section, we give the analysis of the agreement protocol described in section 4, on the $O(\log^* N)$ degree networks described in section 3. We analyze the case in which processors fail with constant probability $\epsilon \geq \frac{\alpha}{N_s}$, where α is the resiliency constant introduced in Theorem 2.1, and N_s is the size of the fully connected networks on the bottom level of the networks. If the failure probability is a smaller constant, the analysis is similar, and we address this case at the end of the section. The modifications for the analysis of the constant degree networks are quite straightforward and will not be elaborated.

In addition to the notations introduced in the previous sections, we use the following definitions.

Definition 5.1 *Let $\beta_i = \frac{\alpha}{\log N_i}$, where α is the resiliency constant introduced in Theorem 2.1.*
A committee $g \in G_s$ (fully connected) is **bad** *if a third of it's processors are bad.*
A committee $g \in G_i$, $0 \leq i < s$ is **bad** *if more than $\beta_i N_i$ of it's subcommittees are bad.*
A committee which is not bad is **good**.

Definition 5.2 *The* **agreement value** *of a good committee $g \in G_s$, after it executes BA, is defined as the common value all correct processors agree on.*
The **agreement value** *of a good committee $g \in G_i$, $0 \leq i < s$ after it executes* **agree**, *is defined as the last agreement value of the majority of it's good subcommittees The subcommittees whose agreement value differs from the majority, are called* **excluded**.

Lemma 5.1 1. *If a committee $g \in G_i$, $0 \leq i \leq s$, is good, and the number of bad subcommittees in g is t, then after each call to* **agree**, *all but $t+1$ of the good subcommittees in g will have the same agreement value. Furthermore, If at least $\frac{2}{3}N_i$ of the good subcommittees (processors) in g have the same initial agreement value, then that will be the agreement value of g.*

2. *If neighbouring committees $g_1, g_2 \in G_i$, $0 < i \leq s$, are good, and g_1's agreement value of the message to be sent to g_2 is M, then after executing* **transmit**(g_1, g_2), *the agreement value of g_2 will be M.*

Proof: We prove the lemma by downward induction on i.
For $i = s$: Since on this level all processors in the same committee are fully connected, and every two neighbouring committees are fully connected, BA achieves (*1*) and (*2*) trivially.
Assume the claim holds for all $j > i$. The induction hypothesis actually means that for every $g \in G_i$, the subcommittees of g act like virtual processors. The good subcommittees behave analogically to correct processors, and the behaviour of the bad subcommittees, like the behaviour of faulty processors, may differ from the protocol. A good $g \in G_i$ is thus a virtual b.c.n of size N_i, including at most $\beta_i N_i$ faulty virtual processors, and Theorem 2.1 can be applied to prove part (*1*) of the lemma. In order to prove part (*2*), we note the following. Since $g_1 (\in G_i)$ is good, all but at most $\beta_i N_i + 1$ of it's good subcommittees have the same agreement value of the message M they are about send to $g_2 (\in G_i)$. After g_1's subcommittees transmit their message to g_2's subcommittees, M is the agreement value of all but at most $2\beta_i N_i + 1$ of the good receiving subcommittees. According to (*1*), by the end of the receiving phase agreement on g_2, that will be the agreement value of g_2. ▮

Lemma 5.2 *If processors fail with probability $\epsilon \geq \frac{\alpha}{N_s}$, and the complete network is good, then all but $O(\epsilon N)$ of the correct processors reach agreement. If all processors hold the same initial value, then that will be the value chosen.*

Proof: If the network is good, then by definition, for every $1 \leq i \leq s$, all good level i committees contribute at most a fraction of $\frac{\alpha}{\log N_i}$ bad level $i+1$ subcommittees. According to lemma 5.1, the fraction of level $i+1$ subcommittees excluded from any agreement, is at most the same. Thus, the fraction of correct processors whose agreement value differs from the majority, is at most twice the sum of fractions of bad committees from all levels, and so the number of correct processors excluded from the agreement is bounded above by

$$2\left(\frac{\alpha}{\log N_0} + \frac{\alpha}{\log N_1} + ... + \frac{\alpha}{\log N_{s-1}}\right) \cdot N < 4\alpha \frac{1}{N_s} N = O(\epsilon N)$$

If all processors begin with the same initial value, then part (*1*) of lemma 5.1 can easily be applied to prove that this will be the value chosen. ▮

It remains to be shown that with very high probability, a network is good.

Lemma 5.3 *There exist constants $m > 1$, and $0 < \Upsilon < 1$, such that if the lowest level committees of a network are of size $N_s \geq m$, and processors fail with probability $\frac{\alpha}{N_s} \leq \epsilon \leq \Upsilon$, the probability that the network is bad, is $2^{-\alpha \frac{N}{\log^2 N}}$.*

Proof: We use the following theorem by Raghavan and Spencer [Ra]. Let $Y_1, ..., Y_n$ be independent Bernoully trials, and let μ be the expected value of their sum. For every $\delta > 0$

$$Pr\left(\sum_{i=1}^{N} Y_i \geq (1+\delta)\mu\right) \leq \left[\frac{e^\delta}{(1+\delta)^{(1+\delta)}}\right]^\mu < \left[\frac{1+\delta}{e}\right]^{-(1+\delta)\mu}$$

Let ϵ_i be the probability that a level i committee $g \in G_i$ is bad. Thus

$$\epsilon_s < \left[\frac{1}{3e \cdot \epsilon}\right]^{-\frac{1}{3}N_s}$$

and for every $0 \leq i < s$

$$\epsilon_i < \left[\frac{\alpha}{e \cdot \log N_i \cdot \epsilon_{i+1}}\right]^{-\alpha \frac{N_i}{\log N_i}}$$

Υ and m should be chosen so that they satisfy the following requirements

$$\left[\frac{\alpha}{e \cdot m \cdot (\frac{1}{3 \cdot e \cdot \Upsilon})^{-\frac{1}{3}m}}\right]^{-\alpha \frac{2^m}{m}} \leq \frac{\alpha}{2 \cdot e} 2^{-m}$$

and

$$2^{-\alpha \frac{2^m}{m}} \leq \frac{\alpha}{2 \cdot e} 2^{-m}$$

(An appropriate choice may be $\Upsilon = \frac{1}{6e}$ and $m = O(\alpha^{-1})$)
Using downward induction on i, it can easily be shown that under the above conditions, for every $1 \leq i \leq s-1$, $\epsilon \leq \Upsilon$

$$\epsilon_i \leq \frac{\alpha}{2 \cdot e \cdot N_i}$$

and the lemma follows. ∎

If $\frac{\alpha}{\log N_0} \leq \epsilon < \frac{\alpha}{N_s}$, an analogical analysis can be done. The definition of a bad committee must be altered in the following fashion. A committee $g \in G_i$, $0 \leq i < s$ will be bad if it includes more than $min(\frac{\alpha}{\log N_i} N_i, \lceil \epsilon N_i \rceil)$ bad subcommittees. The definition of a good committee is thus restricted, so that up to an appropriate level, good committees do not contribute *at all* to the number of excluded processors. As long as $\epsilon \geq \frac{\alpha}{\log N_0}$, the probability that the complete network is bad is bounded above by $2^{-\frac{\alpha N_0}{\log N_0}}$.

References

[BG1] P. Berman and J.A Garay, "Asymptotically Optimal Distributed Consensus", *Proc. 16th ICALP*, 1989, pp. 80-94.

[BG2] P. Berman and J.A Garay, "Fast Consensus in Networks of Bounded Degree", *4th International workshop on Distributed Algorithms*, 1990, pp. 321-333.

[D] D. Dolev, "The Byzantine Generals Strike Again", *J. of Algorithms*, Vol. 3, No.1 (1982), pp. 14-30.

[DPPU] C. Dwork, D. Peleg, N. Pippenger and E. Upfal, "Fault Tolerance in Networks of Bounded Degree", *18th Annual Symposium on Theory of Computing*, 1986, pp. 370-379.

[LPS] A. Lubotzky, R. Phillips, and P. Sarnak, "Ramanugan Conjecture and Explicit Construction of Expanders", *18th Annual Symposium on Theory of Computing*, 1986, pp. 240-246.

[P] N. Pippenger, "On Network of Noisy Gates", *26th Annual Symposium on Foundation of Computer Science*, 1985, pp. 30-38.

[Ra] P. Raghaven, "Probabilistic Construction of Deterministic Algorithms: Approximating Packing Integer Programs", *27th Annual Symposium on Foundation of Computer Science*, 1986, pp. 10-18.

[U] E. Upfal, "Tolerating Linear Number of Faults in Networks of Bounded Degree", To appear in *10th Annual Symposium on Principles of Distributed Computing*, 1992.

Appendix

If processors fail with probability ϵ which is smaller than $\frac{\alpha}{\log N_0}$, we need to slightly modify the network in order to reach $O(\epsilon)$-agreement with very high $(1 - 2^{-\alpha \frac{N}{\log^2 N}})$ probability. This modification includes superimposing a compressor graph on the whole network.

Compressor graphs have the following property [DPPU]:

Lemma 5.4 *If G is a compressor network of size N, then there exist a constant $0 < \theta < 1$ and a compression procedure such that if the set of faulty processors, $\mathcal{T}$, is of size $t \leq \frac{1}{2}\theta N$, and there are $(1-\theta)N$ correct processors which share the same initial value, x, then after applying the compression procedure, at most $t+1$ correct processors will have a value different from x.*

As noted in the previous section, we can reach $O(\epsilon)$-agreement, with probability $(1-2^{-\alpha \frac{N}{\log^2 N}})$, for $\frac{\alpha}{\log N_0} \leq \epsilon < \Upsilon$. If ϵ is smaller, we can still reach $O(\frac{\alpha}{\log N_0})$-agreement with very high probability. All we have to do in order be bring the number of excluded processor down to $O(\epsilon)$, is run the compression procedure using the superimposed compressor communication lines.