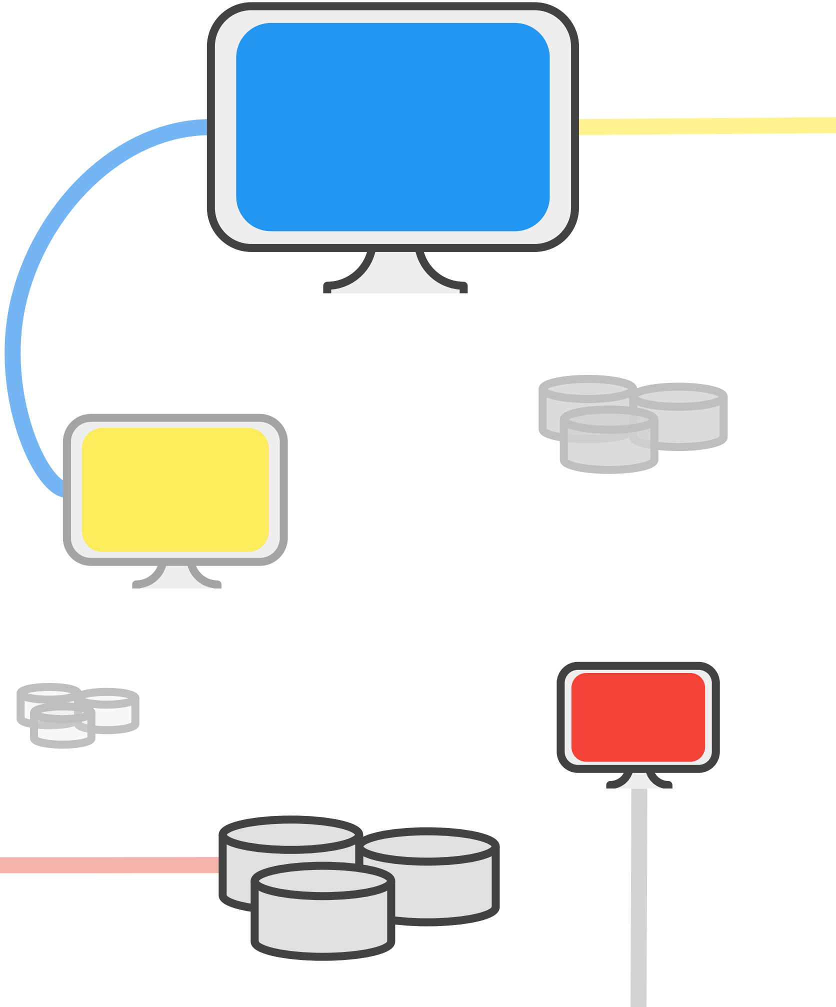
Another advantage of Free Choice: Completely asynchronous agreement protocols

Ben-Or, Michael. "Another advantage of free choice (extended abstract): Completely asynchronous agreement protocols." *Proceedings of the second annual ACM symposium on Principles of distributed computing*. ACM, 1983.

Presented by Fernanda Mora
April, 2016

Contents

- Introduction
- The consensus problem
- A consensus protocol
- Byzantine agreement
- Efficiency



Introduction

Context

- Consensus: termination, agreement and non-triviality
- Fischer, Lynch and Paterson (FLP): no consensus in asynchronous systems with a single undetectable fail.
- Model extensions: randomization, timing assumptions, failure detectors, strong primitives.
- This paper provides the first randomized consensus protocol (not very efficient).
- Idea: allow non-terminating executions but with probability 0.

The consensus problem

Assumptions and problem description

- Completely asynchronous strongly connected system
- Agreement on a {0,1} value
- A process decides by setting a value (cannot be changed).
- Operations: send(Q,m), receive(P)
- Message buffer M to keep undelivered messages
- Configuration: *N* processes states + contents of M
- Step: a single process takes one configuration to another.

Assumptions and problem description

We need to handle probabilities of executions.
In each state of the system there may be a large number
operations of different processes that may occur next.
antes: the choice has been implicit in the choice of a sing
execution, now we want to consider ensembles of many
executions, where the probability of individual exceutions
determined by the return values of the COIN-FLIP opera

- Coin flip operations: returns “random” value.
- Strong adversary-”scheduler”:
  - Controls message system and which process makes a step.
  - Function from partial executions to operations: it chooses which operation is next.
  - Adversary observes the entire history.
- t-correct schedule: in an infinite run with at most t processes make a finite number of steps (t-failures) and any message is delivered if the process makes an infinite number of steps.

A consensus protocol

Overview

Idea: An infinite repetition of rounds to exchange info

- In round r, process P only handles messages with timestamp r
- Each round has two phases: voting and ratification
- In the voting, each process sends a value which is a function of the values collected in the previous round (the first value is the input value)
- In the ratification, each process sends a value which is a function of the collected values

Algorithm

Input: boolean value input
Output: boolean value stored in output
Data: boolean preference, integer round
begin
    preference ← input
    round ← 1
    while *true* **do**

Step 1

        send (1, round, preference) to all processes

Step 2: voting

        wait to receive $n - t$ (1, round, $*$) messages
        if *received more than* $n/2$ *(1, round, v) messages* **then**
            send (2, round, v, *ratify*) to all processes
        else
            send (2, round, ?) to all processes
        end

Step 3: ratification

        wait to receive $n - t$ (2, round, $*$) messages
        if *received a (2, round, v, ratify) message* **then**
            preference ← v
            if *received more than* t *(2, round, v, ratify) messages* **then**
                output ← v
            end
        else
            preference ← `CoinFlip()`
        end

Step 4

        round ← round + 1
    end
end

Algorithm 1: Ben-Or's consensus protocol. Adapted from [20].

Consensus is guaranteed

Theorem 1. Let $N > 2t$. For any t-correct schedule and any initial values of the processes, the above protocol guarantees, with probability 1, that:

(i) all the processes will eventually decide on the same value v

(ii) if all processes start with the value v, then within one round they will all decide v

(iii) if for some round r, some process decides v in step 3, then all other processes will decide v within the next round

Remark: If $N <= 2t$ then consensus is impossible, since the schedule can then simulate a network partition.

Agreement is guaranteed

- At most one value can receive a majority of votes in the first stage of a round, so for any two messages (2;r;v;ratify) and (2;r;v';ratify), v=v'
- If some process sees t+1 (2;r;v;ratify) messages, then every process sees at least one (2;r;v;ratify) message.
- If every process sees a (2;r;v;ratify) message, every process votes for v in the rst stage of round r+ 1 and every process that has not already decided decides v in round r+1

Validity is guaranteed

- Validity follows by a similar argument; if all processes vote for the their
- common input
- v
- in round 1, then all processes send (2
- ;r;v;
- ratify
-) and decide
- in the second stage of round 1

Termination is guaranteed

- At most one value can receive a majority of votes in the first stage of a round, so for any two messages (2;r;v;ratify) and (2;r;v’;ratify), v=v’
- If some process sees t+1 (2;r;v;ratify) messages, then every process sees at least one (2;r;v;ratify) message.
- If every process sees a (2;r;v;ratify) message, every process votes for v in the rst stage of round r+ 1 and every process that has not already decided decides v in round r+1

Byzantine agreement

Description

- Arbitrary behaviour of faulty processes
- A process can determine the originator of a message he received.
- Scheduler:
    - When each process will make a step
    - What the faulty processes do.

Byzantine Protocol

Process P: Initial value x_p.

step 0: set r == 1.

step 1: Send the message (1, r, x_p) to all the processes.

step 2: Wait till messages of type (1, r,*) are received from N-t processes. If more than (N+t)/2 messages have the same value v, then send the message (2, r, v, D) to all processes. Else send the message (2, r, ?) to all processes.

step 3: Wait till messages of type (2, r, *) arrive from N-t processes.

(a) If there are at least t+1 D-messages (2, r, v, D), then set xp :=v.

(b) If there are more than (N + t)/2 D-messages then decide v.

(c) Else set x_p= 1 or 0 each with probability ½

step 4: Set r = r + 1 and go to step 1.

Consensus is guaranteed

Theorem 2. Let $N > 5t$. For any t-correct schedule and any initial values of the processes, the above protocol guarantees, with probability 1, that:

(i) all the processes will eventually decide on the same value v

(ii) if all processes start with the value v, then within one round they will all decide v; and

(iii) if for some round r, some process decides v in step 3, then all other processes will decide v within the next round.

Remark: We do not know whether N>5t is the best possible bound to reach distributed Byzantine agreement.

Efficiency

But termination is slow

- The probability that the algorithm terminates in any given round may be exponentially small as a function of the number of processes, requiring exponentially many rounds.

Theorem 3. If $t = O(N^{1/2})$ then the expected number of rounds to reach agreement in protocols 1 and 2 is constant, (i.e. does not depend on N).

- For deterministic protocols it is known that Byzantine agreement is impossible in less than t+1 rounds of exchange of information.

References

- Ben-Or, Michael. "Another advantage of free choice (extended abstract): Completely asynchronous agreement protocols." *Proceedings of the second annual ACM symposium on Principles of distributed computing*. ACM, 1983.
- *Aspnes, James. "Randomized protocols for asynchronous consensus." Distributed Computing 16.2-3 (2003): 165-175.*

Thanks