Distributed Algorithms
Lecture 9

Lecture by Dr. Gadi Taubenfeld
Typeset by Steven Karas

2017-06-07
Last edited 20:15:52 2017-06-08

Disclaimer These lecture notes are based on the lecture for the course Distributed Algorithms, taught by Dr. Gadi Taubenfeld at IDC Herzliyah in the spring semester of 2017. Sections may be based on the lecture slides and accompanying book written by Dr. Gadi Taubenfeld.

Agenda

- Universality of Consensus

The next homework will be published soon

1 Consensus in Synchronous Message-Passing Networks

Consensus Problem Each process p_i has an input value in_i. Consensus provides two properties:

- Agreement: all non-faulty processes eventually decide on the same value v.
- Crash failure validity: v must be the input of some process
- Byzantine failure validity: v must be the input of some non-faulty process.

Note that solving byzantine consensus basically solves the "blockchain" problem of Bitcoin, etc. According to Gadi, the next two years will show whether or not the business model of cryptocurrencies is valid.

Impossibility of Consensus with one faulty process There is no asynchronous algorithm that solves the consensus problem using send/receive messages in the presence of a single faulty process.

1.1 Synchronous model

In a synchronous model, all processes awake simultaneously. They then send, receive, and process their received messages in each round.

Time Complexity Defined in terms of the number of rounds until the algorithm completes.

Default synchronous model

- Synchronous network
- Number of processes n is a priori known
- Unique identifiers
- Every process is connected to all others (clique)
- Communication by message passing
- Number of failures t is a priori known
- No failures of links or messages
- All processes start simultaneously

Known tight bounds

	crash failures	Byzantine failures
number of processes	$n \geq t+1$	$n \geq 3t+1$
number of rounds	$t+1$	$t-1$

Today We'll show 2 algorithms today:

	crash failures	Byzantine failures
number of processes	$n \geq t+1$	$n \geq 4t+1$
number of rounds	$t+1$	$2(t-1)$

1.2 Crash tolerant algorithm

Algorithm for process p_i with input in_i

```
t+1 times:
  if flag = true:
    send v to all processes
  wait to receive all values
  temp = v
  v = minimum among all received values at this round and current value of v
  flag = v < temp
decide on v
```

Lemma 1 A clean round is one in which no process fails during the round. If a round is a clean round, all processes will have the same output.

Lemma 2 If all processes enter a round with the same input, they will have the same output.

Proof of agreement Because we have $t+1$ rounds, there must be at least one clean round.

Proof of validity We only select v to be the input of some process and never introduce any other values. Therefore, the output must be the input value of some process.

Complexity In the first round, we send n^2 messages. Each process sends a message to n other processes at most $|V|$ times. As such, the algorithm's message complexity is $n^2 \cdot \min(t+1, |V|)$.

Inefficient Variant

Algorithm for process p_i with input in_i

```
t+1 times:
  send v to all processes
  wait to receive all values
  v = minimum among all received values at this round and current value of v
decide on v
```

Other variants If we replace the number of rounds with t, the algorithm is not correct.

If we replace the number of rounds with $n-1$, the algorithm is correct, because all the failed processes must have died.

If we replace the number of rounds with n, the algorithm is correct, trivially.

1.3 Binary Consensus for Byzantine Failure

This presentation is based on "A Simple proof of a simple consensus algorithm" by J. Misra in 1989. A copy of the paper is included with the slides.

Model As above, but assume n is odd, and at most t Byzantine failures where $n \geq 4t+1$. Mark the number of correct processes as s. Note that $s+t=n$, and that $s > 3n/4$ and $t < n/4$.

Algorithm

Algorithm for process p_i with input in_i

```
v = in_i
for k in 1..t+1:
  # part 1
  send v to all processes
  wait to receive all messages

  # part 2
  if i = k:
    v = majority v from messages
    send v to all processes
  else:
    v_2 = value from k

  v' = the most common v from part 1
  if v' received from s messages (including self):
    v = v'
  else:
    v = v_2
```

Lemma 1: clean round A clean round is a round in which all correct processes output v.

Round k is clean if P_k is correct.

Therefore, P_k got v from $> n/2$ processes, half of which may have been byzantine adversaries. Therefore, P_k got v from $> n/4$ correct processes. As such, every process got v from at least $n/4$ processes.

If a process got at least s messages with the same value, it must be v. If a process got less than s of the same message, it must have chosen v.

Lemma 2 If all the correct processes enter a round with the same input, they will end the round with the same output.

We received s messages with the same value v. Therefore we choose this value.

Proof of Agreement As in the previous algorithm, we are guaranteed to have at least 1 clean round. By lemmas 1 and 2, all valid processes must agree on the same value.

Proof of Validity I wasn't listening for when he covered this.

1.4 Impossibility of consensus with $n/3$ faulty Byzantine processes

Lamport coauthored a paper in 1980 regarding hardware failures. The paper was ignored. He rewrote the paper with terminology changes called "The Byzantine Generals Problem" in 1982.

This has gone on to be one of the most cited papers in the field since.

2 Next week

- Barrier synchronization
- Lock-free concurrent queue implementation