Computer Systems

Exercise 7

Last Exercise

1.2 Synchronous Consensus in a Grid - Crash Failures

(2 dimensional grid)

Consider the same network as in Question 1.1. Assume that some of the nodes crashed at the beginning of the algorithm such that any two correct processes are still connected through at least one path of correct processes.

Let l be the length of the longest shortest path between any pair of nodes in the grid; i.e., l is the number of edges between those two nodes which are "farthest away" from each other. If there are no failures, l is the distance between two corners, i.e. $l = w + h$.

a) Modify the algorithm from 1.1**b)** to solve consensus in $l + 1$ rounds with this special type of crash failures. Show that your algorithm works correctly; i.e., a node does not terminate before it learned the initial value of all nodes.

Answer: keep the exact same algorithm as we discussed last time (forward everything, decide when you don't learn anything new in one round, because it means you learned everything).

Correctness: In round i a node receives all values from neighbors of distance i, it will keep receiving new values every round until it has all

Termination: After at most l+1 rounds all information will have propagated through the grid

Last Exercise

1.2 Synchronous Consensus in a Grid - Crash Failures

Consider the same network as in Question 1.1. Assume that some of the nodes crashed at the beginning of the algorithm such that any two correct processes are still connected through at least one path of correct processes.

Let l be the length of the longest shortest path between any pair of nodes in the grid; i.e., l is the number of edges between those two nodes which are "farthest away" from each other. If there are no failures, l is the distance between two corners, i.e. $l = w + h$.

a) Modify the algorithm from 1.1**b)** to solve consensus in $l + 1$ rounds with this special type of crash failures. Show that your algorithm works correctly; i.e., a node does not terminate before it learned the initial value of all nodes.

b) As an adversary you are allowed to crash up to $w + h$ many nodes at the beginning of the algorithm. Let $w = 7, h = 6$. What is the largest l you can achieve?

Optimal if w ≈ h, gives longest distance of about 3(w+h)

Possible also in a general case, gives longest distance of about 2(w+h)

Last Exercise

1.2 Synchronous Consensus in a Grid - Crash Failures

Consider the same network as in Question 1.1. Assume that some of the nodes crashed at the beginning of the algorithm such that any two correct processes are still connected through at least one path of correct processes.

Let l be the length of the longest shortest path between any pair of nodes in the grid; i.e., l is the number of edges between those two nodes which are "farthest away" from each other. If there are no failures, l is the distance between two corners, i.e. $l = w + h$.

a) Modify the algorithm from 1.1**b)** to solve consensus in $l + 1$ rounds with this special type of crash failures. Show that your algorithm works correctly; i.e., a node does not terminate before it learned the initial value of all nodes.

b) As an adversary you are allowed to crash up to $w + h$ many nodes at the beginning of the algorithm. Let $w = 7, h = 6$. What is the largest l you can achieve?

c) Assume that you run the algorithm with any type of crash failures; i.e, nodes can crash at any time during the execution. Show that with such failures the algorithm does not always work correctly anymore, by giving an execution and a failure pattern in which some nodes terminate too early!

u3, w, u1

1. Node x crashes, u1 learns value, w does not

u3, w, u1, u2

2. Node u1 crashes, u2 learns value, v does not

u3, w, u1, u2

3. After 2 rounds, v will have learned the values of all nodes except x. In round 3 the value of x is at node u3, therefore v does not learn anything new and will terminate prematurely

Last Exercise

2.2 Computing the Average Synchronously

In the lecture, we have focused on a class of algorithms which satisfy termination and agreement. In the following, we drop the termination condition and relax the agreement assumption:

Agreement The interval size of the input values of all correct nodes converges to 0.

a) Suggest a simple synchronous algorithm satisfying the agreement property defined above. Use the strategy from Question 2.1**d)**.

Algorithm 2 Simple Synchronous Approximate Agreement

```
Lest x_u be the input value of node u
repeat:
Broadcast x_u
I := all received values x_v without the largest and the smallest f values
Set x_u := mean(I)
```

Last Exercise

2.2 Computing the Average Synchronously

In the lecture, we have focused on a class of algorithms which satisfy termination and agreement. In the following, we drop the termination condition and relax the agreement assumption:

Agreement The interval size of the input values of all correct nodes converges to 0.

a) Suggest a simple synchronous algorithm satisfying the agreement property defined above. Use the strategy from Question 2.1**d)**.

{-3,-2,-1,0,1,2,3}

b) Apply your algorithm to the input from Question 2.1. Compute 3 iterations assuming that byzantine nodes try to prevent the nodes from converging.

[-4, -4, -3, -2, -1, 0, 1, 2, 3] [-4, -3, -2, -1, 0, 1, 2, 3, -4] [-3, -2, -1, 0, 1, 2, 3, 4, 4]

{-1,-1,-1,0,1,1,1}

[-4, -4, -1, -1, -1, 0, 1, 1, 1] [-4, -1, -1, -1, 0, 1, 1, 1, -4] [-1, -1, -1, 0, 1, 1, 1, 4, 4]

{-2/5, -2/5, -2/5, 0, 2/5, 2/5, 2/5}

[-4, -4, -2/5, -2/5, -2/5, 0, 2/5, 2/5, 2/5] [-4, -2/5, -2/5, -2/5, 0, 2/5, 2/5, 2/5, -4] [-2/5, -2/5, -2/5, 0, 2/5, 2/5, 2/5, 4, 4]

{-4/25, -4/25, -4/25, 0, 4/25, 4/25, 4/25}

Last Exercise

2.3 Computing the Average Asynchronously

Consider the algorithm you derived in Question 2.2 in an asynchronous system. Assume that each node broadcasts the current round together with the current input value.

a) Sketch your algorithm in the asynchronous setting.

Algorithm 3 Simple Asynchronous Approximate Agreement

1: Lest x_u be the input value of node u
2: Let $r := 1$ denote the round
3: **repeat:**
4: Broadcast (x_u, r)
5: Wait until received $n - f$ messages of the form (x_v, r)
6: $I :=$ all received values x_v in round r without the largest and the smallest f values
7: Set $x_u := mean(I)$ and $r := r + 1$

Last Exercise

2.3 Computing the Average Asynchronously

Consider the algorithm you derived in Question 2.2 in an asynchronous system. Assume that each node broadcasts the current round together with the current input value.

a) Sketch your algorithm in the asynchronous setting.

b) Show that byzantine nodes can prevent this algorithm from converging if we apply it to the input sequence from Question 2.1.

Last Exercise

2.3 Computing the Average Asynchronously

Consider the algorithm you derived in Question 2.2 in an asynchronous system. Assume that each node broadcasts the current round together with the current input value.

a) Sketch your algorithm in the asynchronous setting.

b) Show that byzantine nodes can prevent this algorithm from converging if we apply it to the input sequence from Question 2.1.

c) What is the largest value of f for which your algorithm will work in the asynchronous system?

c) All local intervals I of the correct nodes can be shown to intersect in at least one value for $f < n/5$: from the $n - f$ correct values, the nodes can hide at most $2f$ too large or too small values. After the removal, the intervals should intersect, i.e. $(n - f) - 2f - 2f = n - 5f > 1$ should be satisfied.

Last Exercise

2.3 Computing the Average Asynchronously

Consider the algorithm you derived in Question 2.2 in an asynchronous system. Assume that each node broadcasts the current round together with the current input value.

a) Sketch your algorithm in the asynchronous setting.

b) Show that byzantine nodes can prevent this algorithm from converging if we apply it to the input sequence from Question 2.1.

c) What is the largest value of f for which your algorithm will work in the asynchronous system?

d) Show that with the chosen bounds on the number of byzantine nodes each of the algorithms will converge.

In every round, all nodes will have a least one common value inside the intervals.
Therefore, the nodes will converge.

Last Exercise

2.3 Computing the Average Asynchronously

Consider the algorithm you derived in Question 2.2 in an asynchronous system. Assume that each node broadcasts the current round together with the current input value.

a) Sketch your algorithm in the asynchronous setting.

b) Show that byzantine nodes can prevent this algorithm from converging if we apply it to the input sequence from Question 2.1.

c) What is the largest value of f for which your algorithm will work in the asynchronous system?

d) Show that with the chosen bounds on the number of byzantine nodes each of the algorithms will converge.

e) Explain how the bounds change in the asynchronous case if FIFO broadcast is used instead of best-effort broadcast or vice versa?

Now the byzantine nodes can only use scheduling to hide values. So now the byzantine nodes can only make the nodes see 2f different values from each side, so $f < n/4$

Consistency Models

- **Linearizability** (implies both others)
- **Sequential Consistency**
- **Quiescent Consistency**

- If you are confused or need an overview:
  - http://coldattic.info/post/88/

Linearizability

- “one global order”
- Linearizability -> put points on a “line”
- Strongest assumption, implies other two

write x = 1 < read x = 1 < write x= 3 < write y = 1 < write x = 2 < read x = 2 < read = 1

Sequential Consistency

- similar as linearizability, but can ”shift” and “squeeze” threads compared to each other
- sequential consistency -> build “sequences”

Sequential Consistency

- similar as linearizability, but can ”shift” and “squeeze” threads compared to each other
- sequential consistency -> build “sequences”

write x = 1 < write x = 2 < write y = 3 < read x = 2 < read y = 2 < read y = 2 < write y = 1

Quiescent Consistency

- synchronizes all threads whenever there is a time when there is no possible execution
- quiescent -> “Quietschente”

write x = 2 < write x = 1 < write y = 1 < ready y = 1 < read x = 1 < read y = 1

Composable (applies to consistency models)

- Definition: If you only look at all operations concerning any object and the execution is consistent, then also the whole execution is consistent
- sequential consistency is not composable
- linearizability is composable
- quiescent consistency is composable

Logical Clocks

- happened before relation „->“ holds:
  - If f < g on the same node
  - Send happens before receive
  - If f -> g and g -> h then f -> h (Transitivity)
- c(a) means timestamp of event a
- **logical clock: if a -> b, then c(a) < c(b)**
- **strong logical clock: if c(a) < c(b), then a -> b** (in addition)

Lamport Clock

https://www.cs.rutgers.edu/~pxk/rutgers/notes/clocks/index.html

Lamport Clock

- Is a logical clock (so if a -> b then c(a) < c(b))
- but the reverse does not hold, so not a strong logical clock

Vector Clock

now vector of clocks

increase own clock for event

Vector Clock

- what does c(a) < c(b) mean now?
  - if all the entries are in a <= b and at least one entry where a < b
- is a logical clock (so if a -> b then c(a) < c(b))
- is also a strong logical clock (if c(a) < c(b) -> a -> b)
  - intuition: because in order to achieve c(a) < c(b), all entries have to be at least as big, so a message from a must have reached b (not necessarily directly) so that b has the right a value

Consistent Snapshot

- Cut
  - prefix of a distributed execution
- Consistent Snapshot
  - a cut for which holds that for every operation g in that cut, if f->g, then also f is there
  - -> if all “connected” preceding operations are included
- with number of consistent snapshots, one can make conclusions about degrees of concurrency in system

NTP

propagation delay = 10

processing time = 5

propagation delay = 10

skew = (15-0) +(20-25)/2 = 5

GPS – General idea

GPS - Problem

- Problem: we do not have the same time as the satellite, so calculating the distance might not be accurate
- Solution: take measurement from forth satellite!

GPS - Refined

Quiz

- Sequential Consistency implies Quiescent consistency
  - *wrong. E.g. x = 1. 5 is a valid outcome for sequential consistency, but not quiescent*

- Are there guarantees a lamport clock can achieve and a vector clock cannot?
  - *No, because the concept of a lamport clock is included in the vector clock concept*
- A high number of consistent snapshot implies a high level of concurrency
  - true

Quiz

- What is the difference between jitter and drift?
  - Shift: Jitter:

Quiz

1.1 Clock Synchronization

a) Assume you run NTP to synchronize speakers in a soccer stadium. Each speaker has a radio downlink to receive digital audio data. However, there is no uplink! You decide to use an acoustic signal transmit by the speaker. To synchronize its clock, a speaker first plays back an acoustic signal. This signal is picked up by the NTP server which responds via radio. The speaker measures the exact time that passes between audio playback and radio downlink response. What is likely the largest source of error?

b) What are strategies to reduce the effect of this error source?

c) Prove or disprove the following statement: If the average local skew is smaller than x, then so is the average global skew.

d) Prove or disprove the following statement: If the average global skew is smaller than x, then so is the average local skew.

Quiz

2.1 Different Consistencies

Prove or disprove the following statements:

a) Neither sequential consistency nor quiescent consistency imply linearizability.

b) If a system has sequential consistency **and** quiescent consistency, it is linearizable.