Dairon Andrés Benites Aldaz
DD2443 HT22 (52168) Parallel and Distributed Computing
October 7, 2022

Lecture 12. Exercise

Question 1. Does the Queen/King algorithm work for non-binary input? Either prove that it does, or exhibit a counterexample.

Yes, it does. It works for any input not just binary input. Even if each node is not given a binary input, to solve consensus, tha algorithm must compute output values at all correct nodes. Also by definition, after broadcast if several values have the same (highest) frequency, we can choose any value e.g. the smallest.

Question 2. Modify the Dolev-Strong algorithm for Byzantine consensus using authentication so that it handles arbitrary input. The processes may also agree on a "sender faulty" value. Give a proof that your algorithm is correct.
This algorithms runs in rounds, and requires clocks to be synchronized between all participating nodes. Each round begins at a constant time-frame in which receivers can pass along a message. The algorithm must run for t +1 number of rounds - where t is the number of adversary or faulty nodes we can have in our network without failing to reach consensus upon a message (t is the constant that describes the basic network security assumption).
listings

```
bool isValid(message, i) {
  set signatures
  if (message.signatures.count != i) {
        return false
  }
  If (message.signatures.contains(origin_sig) != true){
        Return false
  }
  for (sign in message.signatures) {
        if (signatures.contains(sign)) {
            return false
        }
        signatures.add(sign)
  }
  return true
}

void phase(i) {
  for (msg in INCOMING_MESSAGES) {
        if (VALUES.count == 2) {
            return
        }
        if (isValid(msg)) {
            if (message.sender == SENDER) {
                VALUES.add(msg.value)
            }
            sign_and_send_all_but_signed_processors(msg)
        }
  }
  if (i == t+1) {
        if (VALUES.count == 1) {
            return VALUES[0]
        }
        return 0
```

```
    }
}
```

Question 3. Explain why the Ben-Or randomized consensus algorithm does not work when processes just set x = 1 instead of choosing x

randomly having the same initial value, only if all correct processes have that value they will recieve n-2f proposals containing x in the first round and they will decide on x not for every other process. Tus, in the worst case all n-f correct processes need to choose the same bit randomly.

If we are not choosing x randomly with P[X=0] = p[X=1] = 1/2 // for binary input