Distributed Systems
236351
Tutorial 4
Consensus

Dolev Adas
(Revised by Yehonatan Buchnik)

Technion

November 15, 2018
**Consensus**

**Motivation**

Once upon a time, in a faraway kingdom there were two honored princes. The princes were commanded to conquer a rebel city. But, due to the battle-field’s condition, the attack will succeed only if the two princes attack together. What should they do?

- What if they can’t limit the arrival time of the messengers?
- What if the rebels catch the messengers?
- What if the messengers are not loyal?
Consensus

Definition

- In the Consensus problem, every process $p_i$ proposes a value $v_i$ and all correct processes have to decide on some value $v$, in relation to the set of proposed values.

More formally, a distributed consensus algorithm must satisfy:

- **Termination:** every correct process eventually decides some value
- **Validity:** if a process decides $v$, then $v$ was proposed by some process
- **Agreement:** no two correct processes decide differently
Consensus

The FLP Result

It is impossible to solve consensus in an asynchronous fully distributed system even if only one process might crash (the FLP result).

- Why, intuitively, the FLP holds?
- How can it be circumvented?
  - Failure detectors
  - Partial synchrony assumptions
  - Randomization
  - Restricting the set of possible input vectors
Failure Detectors

Unreliable Failure Detectors

- Proposed by Chandra and Toueg
- FDs enrich the asynchronous model with a model of an external failure detection mechanism that can make mistakes
- A distributed failure detector $D$ consists of a local failure detector module $D_p$ at each process $p$
- When $D_p$ suspects a process $j$ to have crashed it adds $j$ to $\text{suspects}_p$
  - if later on $D_p$ realizes it made a mistake, it can remove $j$ from $\text{suspects}_p$
- Failure detectors are defined in terms of abstract properties:
  - Completeness
  - Accuracy
Failure Detectors

Completeness classes

- **Strong Completeness**: Eventually, every process that crashes is permanently suspected by *every* correct process

- **Weak Completeness**: Eventually, every process that crashes is permanently suspected by *some* correct process
Failure Detectors

Accuracy classes

- **Strong Accuracy:** No process is suspected before it crashes
- **Weak Accuracy:** Some correct process is never suspected
- **Eventual Strong Accuracy:** There is a time after which all correct processes are not suspected by any correct process
- **Eventual Weak Accuracy:** There is a time after which some correct process is never suspected by any correct process
## Failure Detectors

<table>
<thead>
<tr>
<th>Completeness</th>
<th>Accuracy</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Strong</td>
<td>Weak</td>
<td>Eventual Strong</td>
</tr>
<tr>
<td>Strong</td>
<td>Perfect</td>
<td>Strong</td>
<td>Eventually Perfect</td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>S</td>
<td>♦P</td>
</tr>
<tr>
<td>Weak</td>
<td>Weak</td>
<td></td>
<td>♦Q</td>
</tr>
<tr>
<td></td>
<td>Q</td>
<td>W</td>
<td>♦W</td>
</tr>
</tbody>
</table>
A distributed algorithm $T_{D \rightarrow D'}$ transforms a failure detector $D$ into a failure detector $D'$, if it implements $D'$ using $D$

$T_{D \rightarrow D'}$ is called a *reduction algorithm* and $D'$ is reducible to $D$

- denoted $D \geq D'$
- $D'$ is weaker than $D$

Is it possible that $D' = D$?
## Failure Detectors

\[ \diamond W \geq \diamond S \]

### Algorithm 1 Code for p

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td><strong>init</strong>: suspected(p) \diamond S \left{ \right} \leftarrow \emptyset</td>
</tr>
<tr>
<td>2.</td>
<td><strong>while</strong> true <strong>do</strong></td>
</tr>
<tr>
<td>3.</td>
<td>suspected(p) \left{ \right} \leftarrow \diamond W</td>
</tr>
<tr>
<td>4.</td>
<td>broadcast(suspected(p))</td>
</tr>
<tr>
<td>5.</td>
<td><strong>end while</strong></td>
</tr>
<tr>
<td>6.</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td><strong>Upon receiving</strong> suspected(q) \text{ from } q \text{ do}</td>
</tr>
<tr>
<td>8.</td>
<td>suspected(p) \diamond S \left{ \right} \leftarrow (\text{suspected}(p) \diamond S \cup \text{suspected}_q) \backslash {q}</td>
</tr>
</tbody>
</table>

### Correctness proof:

- **Eventual Weak Accuracy** is the same in both.
- **Strong Completeness**: Due to the weak completeness of \( \diamond W \), for every crashed process \( c \) there is at least one correct node \( p \) that suspects it. According to the algorithm \( p \) disseminates its suspected list to all, making every correct process eventually suspecting \( c \).
Failure Detectors

\( \Diamond S \geq \Diamond W \)

- What about the opposite direction? does \( \Diamond S \geq \Diamond W \)?
- Conclusion: \( \Diamond S = \Diamond W \)
Consensus Algorithm

Solving Consensus Using \( \diamond S \)

The algorithm parameters:

- \( r \) - round
- \( c \) - coordinator
- \( \text{est} \) - estimation
- \( v \) - value
- \( n \) - number of nodes

Assumptions:

- \( n > 2f \) where \( f \) is the number of faulty nodes
Consensus Algorithm
Solving Consensus Using \( \diamond S \)

**Algorithm 2** Code for \( i \)

1: **init**: \( r_i \leftarrow 0, \text{est}_i \leftarrow v_i \)
2: **while** didn’t decide **do**
3: \( c \leftarrow (r_i \mod n) + 1; \text{est}_c \leftarrow \text{nil}; r_i \leftarrow r_i + 1 \)
4: **if** \( i = c \) **then**
5: \( \text{est}_c \leftarrow \text{ets}_i \)
6: **else**
7: **wait until** \( < \text{EST}, r_i, v > \) is received from \( c \) or \( c \) is suspected
8: **if** \( < \text{EST}, r_i, v > \) is received from \( c \) **then**
9: \( \text{est}_c \leftarrow v \)
10: **end if**
11: **end if**
12: \( \text{broadcast}(< \text{EST}, r_i, \text{est}_c >) \)
13: **wait until** \( < \text{EST}, r_i, \text{est}'_c > \) messages was received from \( n - f \) nodes
14: \( \text{rec}_i \leftarrow \{ \text{est}'_c | < \text{EST}, r_i, \text{est}'_c \text{ was received}> \} \)
15: **if** \( \text{rec}_i = \{ v' \} \) **then**
16: **decide** \( v' \) and \( \text{broadcast}(< \text{DECIDE}, v' >) \)
17: **else if** \( \text{rec}_i = \{ v', \text{nil} \} \) **then**
18: \( \text{est}_i \leftarrow v' \)
19: **end if**
20: **end while**
21: 
22: **Upon receiving** \( < \text{DECIDE}, v' > \) from \( q \) **do**
23: **decide** \( v' \) and \( \text{broadcast}(< \text{DECIDE}, v' >) \)
Correctness proof

Validity

Lemma ($V_1$)

Assume round $r$, node $p$ and coordinator $c$

1. $est_c$ proposed by some node
2. $rec_i \in \{\{est_c\}, \{nil\}, \{est_c, nil\}\}$
3. $est_i$ in the beginning of $r + 1$ proposed by some node

Proof.

For $r = 0$ the claim trivially holds as $est_c = v_c$, $rec_i \in \{\{est_c\}, \{nil\}, \{est_c, nil\}\}$ and hence on $r = 1$, $est_i$ proposed by some node.

Assume the lemma is correct up to round $r$. Then, in round $r + 1$ it is hold that $est_c = est_i^{r-1}$ which was proposed by some node. Hence, following the algorithm sub claims 1 and 2 hold as well.
Correctness proof

Validity

Theorem (Validity)

If a process decides v, then v was proposed by some process.

Proof.

By Lemma V_1, Validity trivially holds.
Correctness proof

Termination

**Lemma** \((T_1)\)

*No correct node is blocked forever while running the algorithm.*

**Proof.**

Following the code, a node may blocked in line 7 or 13. By **strong completeness** of \(\diamond S\) every crashed process eventually became suspected by every correct nodes. Hence, no correct node is blocked forever in line 7. Following the model assumptions, \(n > 2f\), hence no correct node is blocked forever in line 13.
Correctness proof

Termination

**Theorem (Termination)**

*Every correct node eventually decides*

**Proof.**

By Lemma $T_1$ no correct node is being blocked forever while executing the algorithm. By **weak accuracy** of $\Diamond S$, eventually, there is a correct node $q$ which is not suspected by any correct node. Following the code, it comes out that $q$ manages to disseminate its value $v$ to all. $v$ will be broadcast by every correct node and hence the condition in line 15 holds for every correct node.
Correctness proof

Agreement

Theorem (Agreement)

No two correct nodes decide differently.

Proof.

Assume by w.o.c that the theorem does not hold. Then there are two correct nodes $p$ and $q$ such that $p$ decided $v$, $q$ decided $v'$ and $v \neq v'$ and they both decided in round $r$. By the algorithm, that means that both $p$ and $q$ have received from $n - f$ nodes $est_c' = v$ and $est_c'' = v'$ respectively. But, as $n > 2$ it means that least one correct node vote twice which is impossible. Else, if $p$ decided in round $r$ and $q$ did not, then $rec_q = \{v, \text{nil}\} \Rightarrow q$ adopts $v \Rightarrow$ no other value can be decided.